الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Systèmes Informatiques

Thème :

Identification des services web en utilisant des techniques à base de l'IA

Encadré Par :

Mr Ali Seridi

Présenté par :

KHALDI Mabrouk

Juillet 2019

Remerciement

En premier lieu, nous remercions Dieu qui nous a procuré cette réussite.

Nous remercions Monsieur Seridi. Ali mon directeur de projet pour sa disponibilité, sa patience, ses précieux conseils et ses remarques constructives. Aussi pour son aide précieux en termes de documentation et de disponibilité.

Je désire aussi remercier les professeurs de l'université 08 MAI 1945 Guelma (département d'informatique), qui m'ont fourni les outils nécessaires à la réussite de mes études universitaires.

Je tiens à remercier les membres jury qui m'ont fait l'honneur d'évaluer mon travail..

Finalement, un grand merci chaleureux à mes –parents, mes sœurs, mon frère, ma femme, et toute ma famille pour leur soutien constant et leurs encouragements.

Merci tous mes amis d'étude de cette année, Nous remercions également tous ceux qui nous aidé à la réalisation de ce projet.

KHALDI Mabrouk

Département d'Informatique

Université 08 Mai 1945 Guelma

Guelma, Le15/07/2019

Dédicace

Tous les mots ne sauraient exprimer la gratitude, l'amour, le respect, la reconnaissance, c'est tous simplement que : Je dédie ce travail à :

A Ma tendre Mère : Tu représente pour moi la source de tendresse et l'exemple de dévouement qui n'a pas cessé de m'encourager. Tu as fait plus qu'une mère puisse faire pour que ses enfants suivent le bon chemin dans leur vie et leurs études.

A L'esprit de mon père : Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours pour vous. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être. Ce travail est le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation le long de ces années.

A Mes chères sœurs et mon frère: pour vous exprimer mon amour eternel, mes respects et ma fierté d'être m'membre de cette famille.

A Ma très chère femme : Tes sacrifices, ton soutien moral et matériel m'ont permis de réussir mes études. Ce travail soit témoignage de ma reconnaissance et de mon amour sincère et fidèle.

A Monsieur Seridi Ali

A tous les membres de ma promotion.

A tous mes enseignants depuis mes premières années d'études.

A tous ceux qui me sens chers et que j'ai amis de citer.

Tous les mots ne sauraient exprimer la gratitude, l'amour, le respect, la reconnaissance, c'est tous simplement que : Je dédie ce travail à :

A Ma tendre Mère : Tu représente pour moi la source de tendresse et l'exemple de dévouement qui n'a pas cessé de m'encourager. Tu as fait plus qu'une mère puisse faire pour que ses enfants suivent le bon chemin dans leur vie et leurs études.

A L'esprit de mon père : Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours pour vous. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être. Ce travail est le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation le long de ces années.

A Mes chères sœurs et mon frère: pour vous exprimer mon amour eternel, mes respects et ma fierté d'être m'membre de cette famille.

A Ma très chère femme : Tes sacrifices, ton soutien moral et matériel m'ont permis de réussir mes études. Ce travail soit témoignage de ma reconnaissance et de mon amour sincère et fidèle.

A Monsieur Seridi Ali

A tous les membres de ma promotion.

A tous mes enseignants depuis mes premières années d'études.

A tous ceux qui me sens chers et que j'ai amis de citer.

Résumé

Ce mémoire s'intéresse à la réingénierie des application orienté objet vers des application orienté service. Une étape essentielle dans la réingénierie est l'étape de l'identification des services. Pour identifier les services de façon automatique ; nous avons dû modéliser le problème en problème de regroupement (clustering). Le clustering est l'organisation d'un ensemble de données en classes homogènes. Dans notre cas Nous voulons regrouper les classes objets sous forme d'ensemble représentant des services. Nous avons choisi un des algorithmes non supervisés appelé DBSCAN. Il se base sur le regroupement en prenant en compte la densité.

Les mots clé

Architecture Orienté Service, La Réingénierie, La Migration Vers SOA**,** Identification Des Services, Classification Non Supervisée, Algorithme DBSCAN

Sommaire

Chapitre 2 : Les algorithmes de classification non supervisé.

Partie II

Chapitre 3 : Conception.

Chapitre 4 : Implémentation.

Liste des figures

Liste des tableaux

Liste des abréviations

SOA :**S**ervice **O**riented **A**rchitecture.

EAI : Enterprise **A**pplication **I**ntégration

DBSCAN : Density **B**ased **S**patial **C**lustering of **A**pplications with **N**oise

OO: **O**rienté **O**bjet.

OS: **O**rienté **S**ervice.

SI : **S**ystème d'**I**nformation.

CAH : **C**lassification **H**iérarchique **A**scendante.

CHD : **C**lassification **H**iérarchique **D**escendante.

EM : .**E**xpectation **M**aximisation.

SC : **S**oustractive **C**lustering.

UA : **U**sage **A**ttribut.

IM : **I**nvocation **M**éthode.

JSP : **J**ava **S**erver **P**ages.

WEKA: Waikato Environment for Knowledge Analysis

GPL : Group Public License

JDT : **J**ava **D**evelopment **T**ools.

PDE : **P**lug-in **D**éveloppent **E**nvironnent.

IDE :**I**ntegrated **D**evelopment **E**nvironment.

AST :**Abstract S**yntax **T**ree.

JAR : **J**ava **Ar**chive.

Introduction générale

Plusieurs technologies innovantes ont été introduites ces dernières années, pour renforcer la communication et le partage des ressources. Parmi ces dernières on trouve l'architecture orientée services est plus communément connue sous le sigle SOA (Services *Oriented Architecture).*

L'objectif de la (SOA) est d'intégrer les briques ou composants logiciels au sein du SI de l'entreprise. Cette méthode s'attache à décomposer chaque composant en un ensemble de fonctions basiques appelées **services**. L'objectif étant de pouvoir réutiliser simplement ses services. Les utilisateurs de ceux-ci peuvent être d'autres services, des logiciels de l'entreprise, des applications externes, etc….

Notre étude s'inscrit dans le cadre de la réingénierie des systèmes patrimoniaux vers des nouvelles architectures qui est la SOA. Nous avons proposé de transformer les applications orientées objet vers des applications orientées service afin de bénéficier des avantages de cette nouvelle architecture.

La problématique étudiée est comment automatiser la phase de l'identification de service à partir d'applications orientée objet. Nous avons exploré les algorithmes de classification non supervisés, à savoir (DBSCAN) pour regrouper l'ensemble des classes homogènes pour former un service tout en respectant les caractéristiques de la SOA. Nous avons commencé par étudier les différents algorithmes existants afin de choisir celui qui convient à notre cas. Le choix a été fait sur l'algorithme DBSCAN.

Notre mémoire est structurée en deux parties et organisée de la manière suivante :

La première partie : est consacré à un état de l'art et composé de deux chapitres, le premier sur la réingénierie et migration vers SOA où nous parcourons brièvement les concepts de la SOA, de la réingénierie et de la migration des services vers la SOA. Dans le deuxième chapitre nous explorons les différents algorithmes de classification non supervisées tel que DBSCAN.

La deuxième partie : est composée de deux chapitres ; le premier chapitre c'est la conception de notre projet, dans cette étape nous allons expliquer le processus que nous avons suivi pour concevoir notre projet et comment nous avons adapté l'algorithme choisi afin d'identifier les services. Le deuxième chapitre c'est l'implémentation, dans ce dernier nous définirons les

outils, l'interface et les résultats obtenus. Nous terminons par une conclusion générale et quelques perspectives.

Partie I

Chapitre1

Réingénierie et Migration vers SOA

1 Introduction :

Nous vivons actuellement dans une société de consommation orientée vers la technologie. Cette dernière est présente dans la vie de tous les jours autant pour les particuliers que pour les entreprises. Un des concepts clé de cette mutation technologique est l'informatique, qui est devenu un des piliers d'une bonne gestion organisationnelle [1].

Depuis quelques années, sur les problématiques autour de la conception de l'architecture logicielle, la notion d'Architecture Orientée Services (SOA) s'est largement imposé grâce en particulier à la montée en puissance de l'Internet. Maintenant, le concept de SOA recouvre des aspects et des domaines très variés qui dépassent d'une certaine manière largement le domaine initial qu'est l'architecture logicielle. [w2]

L'architecture orientée service ou SOA (Service Oriented Architecture) est aujourd'hui envisagée par de nombreuses entreprises dans le cadre de l'évolution de leur système d'information.

A travers ce chapitre nous essayerons de présenter les notions relatives à la SOA et réingénierie et les techniques de migration des systèmes patrimoniaux vers la SOA.

2. L'Architecture Orientée Services

La SOA (Service Oriented Architecture - ou Architecture Orientée Services) est souvent assimilé à des technologies mais ce sont en réalité des principes d'architectures. En effet, la notion de SOA renvoie à une nouvelle manière d'intégrer et de manipuler les différentes briques et composants applicatifs d'un système informatique (comptabilité, gestion de la relation client, production, etc.) et de gérer les liens qu'ils entretiennent. [w3].

2.1 Notions de base :

2.1.1 Le service :

2.1.1.1 Définition :

On distingue plusieurs définitions pour le service :

Définition 1 : Un Service est un composant logiciel distribué, exposant les fonctionnalités à forte valeur ajoutée d'un domaine métier [w4].

Définition 2 : Un service est une ressource abstraite qui effectue une tâche cohérente et fonctionnelle [2].

Définition 3 : Un service est considéré comme un processus qui a une interface ouverte, et une granularité grossière. Il peut être facilement décomposé et composé pour mettre en œuvre divers flux de travail [3].

2.1.1.2 Les caractéristiques des services :

Chaque service est caractérisé par les caractéristiques suivantes [w5] :

- **Contrat standardisé :** L'ensemble des services d'un même système technique sont exposés au travers des contrats respectant les mêmes règles de standardisation (l'interface) [w4].
- **Couplage lâche :** Le couplage faible (loosely-coupled) est une propriété qui montre le degré de liaison des services. Les services sont connectés aux autres services ou aux clients à travers des documents standards, ces documents sont réalisés en XML de la même manière que les web services et ils assurent le découplage ou autrement dit la réduction des dépendances [w4].
- **Abstraction :** Le contrat d'un service ne doit contenir que les informations essentielles à son invocation. Seules ces informations doivent être publiées [w4].
- **Réutilisabilité:** Les services doivent encapsuler une logique de traitement suffisamment générique pour être utilisés dans des contextes d'utilisation différents [w4].
- **Autonomie:** Un service doit être totalement autonome. On doit pouvoir le remplacer ou le déplacer sans que cela affecte d'autres services. Un service implémente ses propres composants et ses propres méthodes d'accès aux données, il ne doit dépendre d'aucun élément externe [w5].
- **Sans état (stateless):** Son exécution ne dépend pas de l'invocation préalable d'autres services, l'ensemble du contexte qui est fourni lors de l'invocation par le consommateur suffit au service pour exécuter ses actions [w5].
- **Découvrabilité:** Un service est complété par un ensemble de métas donnés de communication au travers desquelles il peut être découvert et interprété de façon effective [w6].
- **Composabilité:** Un service doit être conçu de façon à participer à des compositions de services [w4].
- **Synchrone ou asynchrone :** Attente de réponse après l'utilisation d'un service ou non [w4].

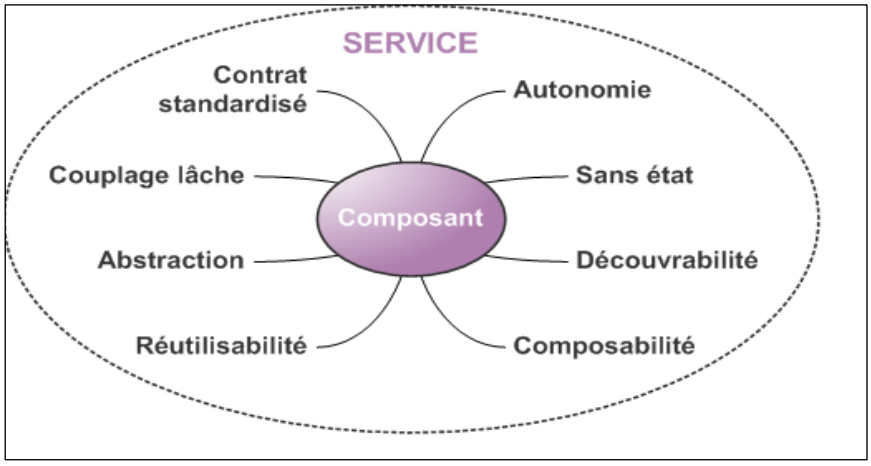


Figure 1.1 : *Les caractéristiques de service [w4].*

2.2 L'architecture orientée service :

2.2.1 Définition :

Une architecture orienté service est une architecture logicielle s'appuyant sur un ensemble de services simples [w1].

La SOA est un ensemble de principes architecturaux qui permettent de développer des systèmes modulaires basés sur des « services » ou des unités de fonctionnalités informatiques. Ces services, qu'ils soient métier ou techniques, sont offerts par une entité, le prestataire de services, et consommés par une autre [4].

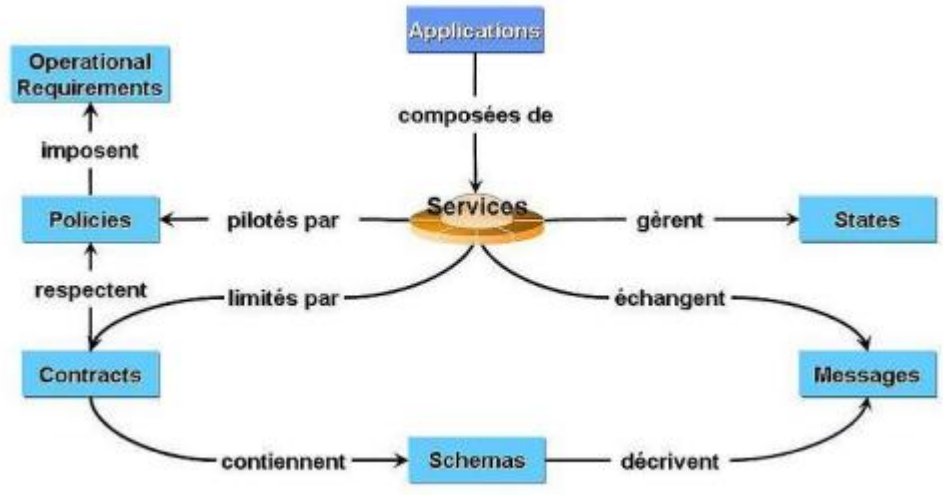


Figure 1.2: *Les concepts de SOA [3].*

Les services au centre de ce schéma sont les unités atomiques d'une architecture orientée services, ces services communiquent entre eux par le biais de messages, ces communications

peuvent être synchrones ou asynchrones. Le langage de programmation du service n'a aucune importante, ainsi que la plateforme d'exécution, matérielle et logicielle. Ces services sont limités par un ensemble de contrats, c'est-à-dire qu'un service doit respecter un ensemble de règle de fonctionnement. [3]

2.2.2 Les caractéristiques de l'architecture orientée service :

Une des plus grandes fonctionnalités des SOA est la réutilisation des services pour plusieurs entreprises. En effet, en étudiant l'évolution des différentes architectures, on peut observer une réutilisation croissante. Ceci était impossible dans le modèle des composants. En passant du procédural à l'orienté objet, on a la notion de réutilisation de classes d'objet. Celle-ci n'a cessé d'augmenter avec la vision orienté composant puis enfin service, voir figure 1.3 [w1]

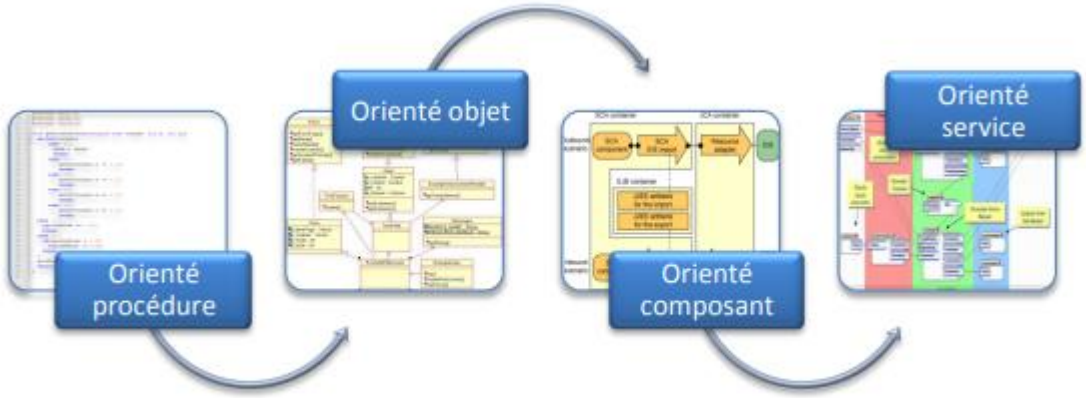


Figure 1.3 : *Cycle de vie des services [3].*

On peut situer les caractéristiques suivantes [w7]:

- **Réutilisation et composition :** Ceci est particulièrement puissant pour créer de nouveaux processus d'affaires rapide et fiable.
- **Recomposition :** La capacité de modifier les processus d'affaires existants ou d'autres applications basées sur l'agrégation des services.
- **La capacité à progressivement changer le système :** Commutation des prestataires de services, l'extension des services, en modifiant les fournisseurs de services et les consommateurs. Tous ces éléments peuvent être faits en toute sécurité, grâce au couplement bien contrôlé.
- **La capacité à construire progressivement le système :** Cela est particulièrement vrai pour toute intégration basée sur SOA.

2.2.3 Les acteurs de l'architecture orientée service :

L'architecture orientée service se représente en faisant intervenir trois acteurs : le consommateur, le service, et le répertoire de services.

Le consommateur correspond à l'application cliente (ou à un autre service), qui fait appel au service pour une tâche précise. Ce consommateur trouvera les informations à propos du client au sein du répertoire de services, où sont enregistrés et triés un grand nombre de ceux-ci. [w8]

Un répertoire peut être privé, c'est-à-dire interne à l'entreprise, ou public. Le service fournit une fonctionnalité bien définie et répond à trois fonctionnalités caractéristiques : il est indépendant, il peut être découvert et appelé de manière dynamique et il fonctionne seul. [w8]

Le répertoire de services a un rôle primordial dans la SOA. C'est lui qui reçoit la requête du consommateur, lui qui découvrira le service idoine, et lui qui agira en tant que proxy (intermédiaire) entre consommateur et service. En s'assurant que les fournisseurs de services informent régulièrement les répertoires de leurs nouveautés, le consommateur peut constamment profiter de celles-ci sans pour autant devoir mettre à jour ses méthodes [w8].

2.2.4. Déploiement de la SOA :

SOA avec sa nature faiblement couplée permet aux entreprises ce qui suit :

- Le déploiement de la SOA peut-être effectué sur des serveurs d'intégrations, EAI (Enterprise Application Intégration). Ces serveurs organisent la circulation de l'information entre les applications et les rendent interopérables en développant des connecteurs. [w1]
- Brancher de nouveaux services.
- Améliorer les services existants de façon granulaire pour répondre aux nouveaux besoins.
- Offrir la possibilité de rendre les services consommables à travers différents canaux.
- Exposer les applications existantes de l'entreprise comme des services, tout en préservant les investissements d'infrastructure informatique existante.[w7]

2.2.5 Avantages et inconvénients de la SOA

2.2.5.1 Les avantages de l'architecture orienté service :[w7]

- Une modularité permettant de remplacer facilement un composant ou service par un autre.
- Une réutilisabilité possible des composants par opposition d'un système tout en un fait sur mesure pour une organisation.
- De meilleures possibilités d'évolution, ici il suffit de faire évoluer un service ou d'ajouter un nouveau service.
- Une plus grande tolérance aux pannes.
- Une maintenance facilitée.
- Obligation d'avoir une modélisation poussée.
- Possibilité de découpler les accès aux traitements.
- Localisations et interfaçage transparents.
- Possibilité de mise en place facilitée à partir d'une application objet existante.
- Réduction des coûts en phase de la maintenance et d'évolution.
- Facilité d'amélioration des performances pour des applications importantes (répartition de traitements facilités).

2.2.5.2 Les inconvénients de l'architecture orientée service : [w9][w10]

- Nécessité d'appréhender de nouvelles technologies.
- Performances réduites pour les traitements simples (couche supplémentaire).
- Tendance à la multiplication des couches et des messages.
- Problème de coordination ou orchestration entre les divers services.
- Problème de sécurité (plus grande surface exposée).
- Problème de réelle interopérabilité.
- On constate une certaine lourdeur et de la complexité.

3 Réingénierie et migration vers SOA

La réingénierie logicielle est l'approche qui permet l'adaptation du logiciel traditionnel, puisqu'elle est basée sur la remise en cause fondamentale et la préconception radicale des processus organisationnels, afin de réaliser des améliorations spectaculaires des performances courantes [w11].

3.1 Réingénierie

3.1 Définition :

La réingénierie consiste à reconcevoir, repenser ce qui a été conçu dans une démarche d'ingénierie. Il est difficile, voire impossible de faire de la réingénierie si ce qui a été conçu précédemment n'a suivi aucune règle, aucune norme.

Dans le secteur de l'informatique il existe plusieurs types de réingénierie :

- Réingénierie des systèmes d'informations.
- Réingénierie des processus.
- Réingénierie logicielle [w12].

3.1.1 La réingénierie des systèmes d'information :

Il s'agit d'une remise à plat de tout ou partie d'un système d'information, afin d'atteindre de meilleurs niveaux de performances globales. Cela peut par exemple consister à transformer une architecture informatique initialement tournée vers la production en un dispositif orienté vers le client [w13].

3.1.2 La réingénierie des processus :

La réingénierie des processus traite la transformation des processus de l'entreprise, et ce, essentiellement au sein des grandes entreprises. Cette transformation est toujours le fruit d'un processus ou d'un plan de gestion du changement maîtrisé, dans le cadre duquel une organisation audite et compare les processus existants puis tente de les optimiser [w14].

3.1.3 La réingénierie des logiciels :

La mission de la réingénierie des logiciels est l'amélioration et la transformation d'un logiciel existant de telle façon que la compréhension, le contrôle et l'utilisation de ce dernier puissent de nouveau être garantis. La demande en réingénierie de logiciels a connu une croissance spectaculaire liée à l'obsolescence des systèmes de logiciels patrimoniaux, l'obsolescence portant non seulement sur leur architecture, mais également sur les plateformes les supportant ainsi que sur leur adaptabilité et leur stabilité face aux développements techniques requis par l'évolution des besoins. [w15].

La réingénierie des logiciels s'avère nécessaire afin de rétablir et de réutiliser les ressources des logiciels existants, de contrôler davantage les coûts élevés de maintenance des logiciels et de créer une base solide à même de supporter les évolutions à venir [w15].

3.1.3.1 Les étapes de la réingénierie des logiciels :

Les grandes étapes de la réingénierie de logiciel sont :

- **La rétro-ingénierie (reverse engineering):**

C'est le processus d'analyse d'un système en vue d'identifier ses composantes, leurs corrélations et de créer des représentations du système sous une autre forme ou d'un niveau plus élevé d'abstraction.

Dans cette étape, il existe un sous-ensemble qui permet de comprendre la fonction d'un programme grâce à l'interprétation de code et de documentation de l'application, ce sous-ensemble s'appelle rétro-conception (Design Recovery) [5].

- **La recomposition (Forward Engineering):**

C'est le processus qui permet la conception physique d'un système informatique àpartir de sa définition logique c'est à dire le haut niveau d'abstraction [5].

- **La ré documentation (Redocumentation):**

Est la création ou la révision d'une représentation sémantiquement équivalente dans le même niveau relatif d'abstraction. Les formes de représentation résultantes sont habituellement considérées des vues alternatives (par exemple, flux de données, structures de données, et flux d'informations) destinées à une compréhension humaine [w16].

- **La restructuration (Restructuring):**

Est la transformation d'un logiciel en un autre logiciel ayant le même niveau d'abstraction tout en préservant le comportement externe du logiciel (fonctionnalité et sémantique) [w16].

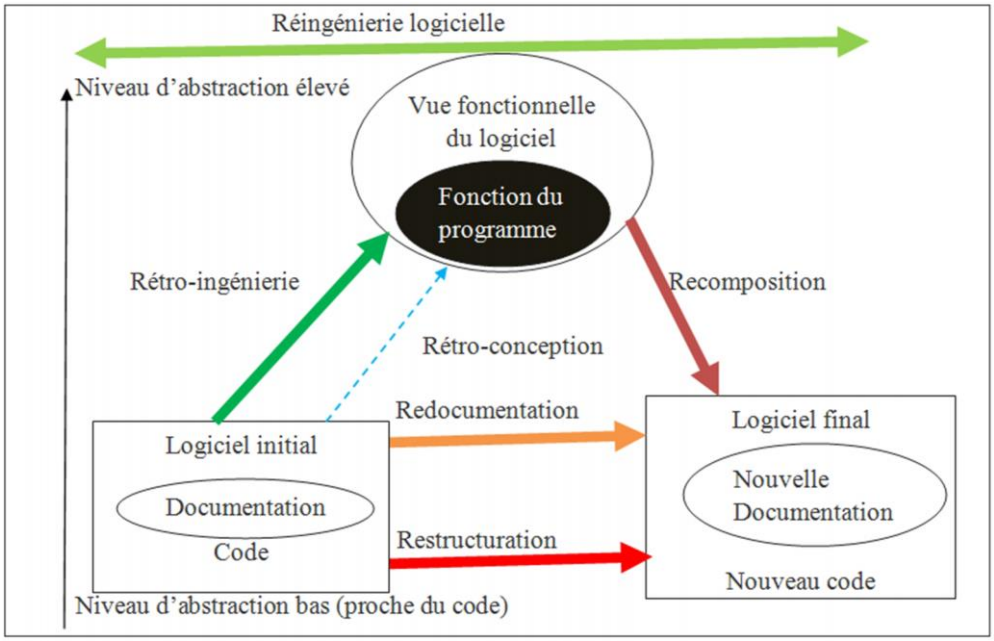


Figure 1.4 : *Les étapes de la réingénierie des logiciels [5].*

3.2 Les avantages de la réingénierie :

Les avantages de la réingénierie sont :

- Réduire les risques : redévelopper un système critique introduit beaucoup de nouveaux risques.
- Réduire les coûts : redévelopper un nouveau système coûte plus cher que la modification d'un système existant [w17].
- Réduire la perte de données et la fuite d'informations.
- Améliorer l'efficacité opérationnelle et le service à la clientèle [w17].

3.3 Les risques reliés à la réingénierie :

On désigne plusieurs facteurs de risque lié à la réingénierie :

- La capacité à éliminer ou à réduire les propriétés indésirables du système existant.
- L'introduction des nouvelles technologies n'est pas supportée par les programmeurs.
- L'impact de l'introduction des nouvelles parties au système existant.
- L'impact des changements sur la performance et la robustesse du système existant.

4 La migration vers SOA

La migration ou la modernisation logicielle implique des changements plus étendus qu'une simple maintenance, mais une partie significative du système est conservée. Ces changements

incluent souvent la restructuration du système, l'amélioration des fonctionnalités ou la modification des attributs du logiciel [6].

4.1 Stratégies d'évolution vers une architecture orientée service :

Plusieurs travaux ont été proposés pour faire évoluer les systèmes patrimoniaux, vers une architecture SOA, la majorité tourne autour de la réingénierie. Différentes classifications ont été discutées dans. Puis une nouvelle classification qui répartit les différentes stratégies de migrations vers SOA en trois catégories sa été proposé à savoir [7] :

4.1.1 Approches décisionnelles :

Ce sont des approches qui aident les entreprises à prendre des décisions, après étude des coûts, du contexte, de la faisabilité pour décider est ce qu'il est rentable ou faisable de migrer, d'intégrer, de remplacer ou de ne rien faire. Puis si on opte pour une des solutions, comment l'appliquer, quelles techniques utiliser. La critique qui peut être faite à cette catégorie, c'est qu'elle ne comporte pas de phase implémentation et reste finalement une approche d'aide à la décision [7].

4.1.2 Approches partielles :

Elle englobe des approches qui proposent des solutions partielles en utilisant les techniques de réingénierie pour analyser et récupérer l'architecture et les fonctionnalités du système patrimonial pour enfin avoir comme résultat un nouveau système orienté service, en utilisant l'intégration ou la migration. Mais sans faire une analyse préalable de la faisabilité de l'évolution [7].

4.1.3 Approches techniques :

Elles se basent sur l'aspect technique tel que les méthodes d'analyse de code, d'identification et d'extraction des services, comment les intégrer avec le SP (Système Patrimonial), comment tester les nouveaux services,….etc [7].

4.2 Les approches de migration vers SOA:

Au niveau des approches, il existe trois angles différents pour la mise en place d'une Architecture orientée services qu'on va présenter dans le paragraphe suivant. [8].

- **L'approche bottom up:**

Le premier angle d'attaque part de la couche technique de l'entreprise. Il s'agit dans ce cas d'extraire les services à partir des applications existantes. Le retour de cette démarche est

assez clair et bien établi. On aboutit à un ensemble de services techniques dont le processus d'identification est assez facile, avec par contre un niveau de granularité assez fin et une profusion de service trop importante ce qui engendre une grande difficulté pour s'aligner avec le métier.

- **L'approche top down:**

Le deuxième angle d'attaque est relié directement au métier de l'entreprise. Il consiste à identifier des services réutilisables en se basant sur les descriptions du niveau métier de l'entreprise. Le résultat est un ensemble des services réutilisables de point de vue métier. Cependant, cette méthode néglige l'aspect architectural de service. En effet, un service est une notion d'architecture et le fait de définir des services à partir du niveau métier n'implique pas forcément que ces derniers peuvent être projetés sur le niveau technique afin d'assurer leur fonctionnement.

Les méthodes top-down et bottom-up ne nous permettent pas d'aboutir à une identification de services fiables répondant à la finalité de l'entreprise orientée service. La première aboutit à un ensemble de services très abstraits avec aucune garantie pour leur réalisation du point de vue technologique, quant à la deuxième, elle concourt à un ensemble des services très techniques qui n'implique pas forcément un alignement avec les besoins métier.

- **L'approche meet in the middle:**

Le troisième angle d'attaque est le "meet in the middle" qui mène en parallèle une approche top down pour définir des services de haut niveau nécessaires à la réalisation des processus métiers, ainsi qu'une approche bottom-up dans le but de cartographier l'existant applicatif de l'entreprise pour supporter les services métiers. Ensuite, un croisement entre les deux résultats est fait afin de déterminer comment seront réalisés les processus métiers. Cette démarche paraît la plus intéressante et la plus concrète[8].

4.3 Les bénéfices de la migration :

La migration vers la SOA offre plusieurs bénéfices aux entreprises, parmi ces bénéfices nous citons :

- L'adaptation aux nouveaux besoins.
- L'amélioration des services clients.
- Une intégration plus étroite avec les partenaires et les fournisseurs.
- La réduction du coût d'usage des systèmes d'information.

- L'amélioration de la qualité des données.
- L'amélioration du contrôle et de la gestion de sécurité.
- L'augmentation de la flexibilité et la réactivité des systèmes d'information [9].

4.4 Les difficultés de la migration:

Les efforts de la modernisation des systèmes d'information des entreprises ont échoué dans la plupart du temps à cause de plusieurs facteurs, nous citons les suivants :

- **La complexité des systèmes patrimoniaux:** la complexité est considérée comme le plus grand limiteur du processus de migration, elle est produite à cause de la taille énorme de système, de l'incompréhensibilité des systèmes à migrer et des phases successives de maintenance.
- **Les risques de migration:** les risques de migration ne sont pas majeurs, il est possible d'accepter un certain risque si nous devons accomplir une tâche de migration. Malheureusement, beaucoup d'entreprises sont incapables ou ne veulent pas contrôler le risque correctement. Ceci peut également provenir de l'insuffisance de compréhension de la gestion des risques et des techniques de réduction du risque [9].

5 Conclusion:

Dans ce chapitre nous avons illustré les concepts de base de la SOA qui est un style architectural moderne qui suit des principes de modélisation modernes permettant la réutilisation des services ainsi que l'exécution des processus d'entreprise.

La réingénierie s'applique aux systèmes d'information, les processus des entreprises ainsi que les logiciels afin de les améliorer. Et la migration vers une architecture SOA permet de garder et réutiliser les parties intéressantes qui supportent les besoins métiers du système intégré et en même temps d'accompagner l'évolution technologique

Ils existent Plusieurs stratégies et approches dans la littérature, elles comportent toutes une étape commune et essentielle dans le processus de migration qui est l'identification des services.

L'identification peut se faire de façon manuelle ou automatique. Dans notre travail, nous avons essayé de focaliser notre effort pour automatiser cette phase d'identification.

Le prochain chapitre a pour objectif de présenter une étude sur quelques algorithmes qui nous permettent d'automatiser l'identification des services.

Chapitre2

Les algorithmes de classification non supervisées

1 Introduction:

Il est claire que le processus générale que la classification dans le domaine informatique essaie de l'appliquer sur des données numériques (points, tableaux, images, sons, . . .etc.) , et que le travail général des méthodes de classification , depuis 1749, consiste à imiter et automatiser ce principe en utilisant et inventant des moyens adéquats (matériaux-calculateurs- , et des théories classificatoires. . .etc.).

Allons de ce principe, nous présenterons dans ce chapitre tout d'abord ce que c'est la classification, ses types, ses méthodes, techniques, et Quelques algorithmes de classification non supervisée, . . . etc. et on détaillera à la fin une de ses grandes approches en étudiant et analysant ses algorithmes.

Nous avons dû chercher parmi les algorithmes existants celui qui convient le mieux à notre objectif (dans notre cas l'algorithmes DBSCAN).[w18].

2 La classification:

La classification joue un rôle important dans toutes les sciences et techniques qui font appel à la statistique multidimensionnelle. Citons tout d'abord les sciences biologiques : botanique, zoologie, écologie,...Ces sciences utilisent également le terme de "taxinomie" pour désigner l'art de la classification. De même les sciences de la terre et des eaux : géologie, pédologie, géographie, étude des pollutions, font grand usage de classifications. Elle consiste à regrouper les objets d'un ensemble de données X de nature quelconque en un nombre restreint de classes homogènes.[w19]

Il existe deux types de classification : la classification supervisée et la classification non supervisé. Nous parlerons de classification non supervisée, ou regroupement, lorsque l'on ne dispose d'aucune information à priori sur les objets à traiter, et de classification supervisée, ou classification tout court, dans le cas contraire. Il arrive parfois, que l'information disponible soit incomplète, et on parle alors d'environnement partiellement supervisé.[10]

2.1 Définition:

La Classification est une tâche ou une série de méthodes qu'une théorie unifiée pour pouvoir utiliser les analyses complémentaires, il est souvent important de traduire l'information de fréquence en information thématique [w19].

C'est une discipline scientifique qui trouve dans un ensemble d'objet des groupes homogènes (classes) et bien distincts les uns des autres.

2.2 Les types de classification:

On a généralement le choix entre deux approches : la classification supervisée et non supervisée

2.2.1 La classification supervisée:

La classification supervisée (appelée aussi classement ou classification inductive) a pour objectif « d'apprendre »Consiste à définir une fonction qui attribue une ou plusieurs classes à chaque donnée. Dans cette approche on suppose qu'un expert fournit au par avant les étiquettes pour chaque donnée, les étiquettes sont des classes d'appartenance [w20].

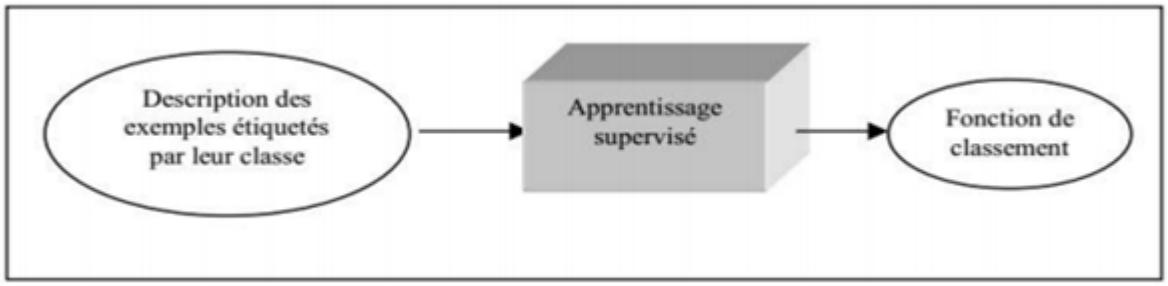


Figure 2.1 : *Classification supervisée .*

2.2.2 La classification non supervisée:

La classification non supervisée (clustering) encore appelé apprentissage à partir d'observations ou découverte, consiste à déterminer une classification« sensée » à partir d'un ensemble d'objets ou de situations données (des exemples non étiquetés), ou bien est un outil très performant pour la détection automatique de sous-groupes pertinents (ou clusters) dans un jeu de données ou d'objet.

Les membres d'un même cluster doivent êtres similaires entre eux, contrairement aux membres des clusters différents (homogénéité interne et séparation externe).[w20] [w21]

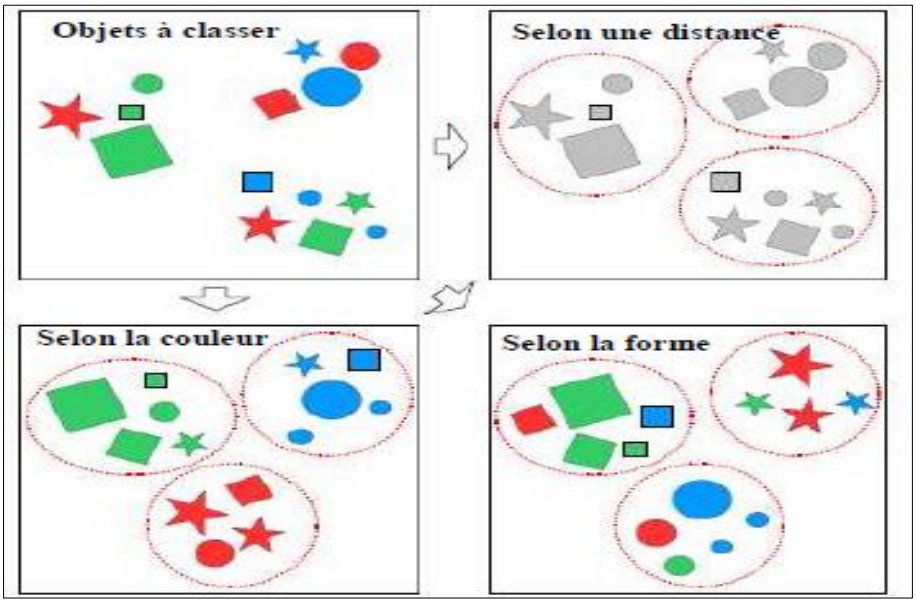


Figure 2.2 : *Classification non supervisée selon différents critères de similarité [11].*

2.3 La similarité et la dissimilarité:

Un calcul de proximité peut mesurer la similarité ou la dissimilarité : plus deux objets se ressemblent, plus leur similarité est grande et plus leur dissimilarité est petite.

La ressemblance peut être mesurée par la distance qui existe entre deux objets ou par la relation entre les attributs de ces objets. Dans la similarité, et selon le type et la représentation des objets, on retrouve plusieurs distances.[12]

Dans ce qui suit, nous présentons quelques-unes entre deux objets d(x1 ; x2). [13].

Distance de Minkowski:

$$d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^q \right)^{\frac{1}{q}}$$

Distance euclidienne:

$$d(x,y) = \sqrt{\left(\sum_{i=1}^{n} |x_i - y_i|^2 \right)} \text{ (Minkowski, } q = 2)$$

Distance de Manhattan:

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i| \text{ (Minkowski, } q = 1)$$

Distance maximum:

$$d(x,y) = \max_{i=1}^{n} |x_i - y_i| \text{ (Minkowski, } q \to \infty)$$

Les distances citées ci-dessus sont exposées en détail dans [13].

3 Quelques algorithmes de classification non supervisée:

Les méthodes de classification non supervisée peuvent être regroupées en cinq grandes catégories : les méthodes hiérarchiques, les méthodes de clustering par partitionnement, les méthodes basées sur la densité des objets dans leur espace de représentation, les méthodes basées sur le réseau de neurone et les méthodes probabilistes. Cette taxonomie est présentée dans le tableau suivant [13][14]:

Les approches de classification non supervisée	Les algorithmes
Approche hiérarchique.	Classification hiérarchique ascendante CAH, Classification hiérarchique descendante CURE, BIRCH, l'algorithme de Ward, etc.
Approche par partitionnement.	IsoData, Fuzzy C-Means, Fast Global K-Means, K-Means++, etc.
Approche de clustering basé sur la densité.	Soustractive clustering, Denclust, Mean-scihft, Density based spatial clustering of applications with noise (DBSCAN), etc.
Approche de clustering basé sur le réseau de neurone.	Kohonen, Adaptive Resonance, etc.
Approche de clusteringprobabiliste.	Expectation maximisation (EM).

Tableau 2.1: *Taxonomie des méthodes de clustering [13][14].*

Dans ce qui suit nous allons présenter quelques algorithmes de ces catégories.

3.1 L'algorithme hiérarchique:

La méthode hiérarchique se présente comme la succession des partitions emboîtées. Elle consiste à calculer une matrice exprimant les distances mutuelles entre les points à classer,

puis, selon le type de l'hiérarchiquement choisis (ascendant ou descendant) soit on suppose chaque individué comme un cluster et on fusionne à chaque étape les deux clusters les plus proches jusqu'à l'obtention d'un seul cluster. Ou bien on procède de façon inverse. On considère l'ensemble des données comme un gros cluster unique, et le scinde en deux clusters "descendants", tel que la distance entre les deux clusters soit le plus grande possible jusqu'à ce qu'il ne reste plus que des clusters qui contient un seul individu [15]

Le résultat de cette méthode est un arbre de partition successive appelé dendrogramme [w20], comme celui de la figure 2.3.

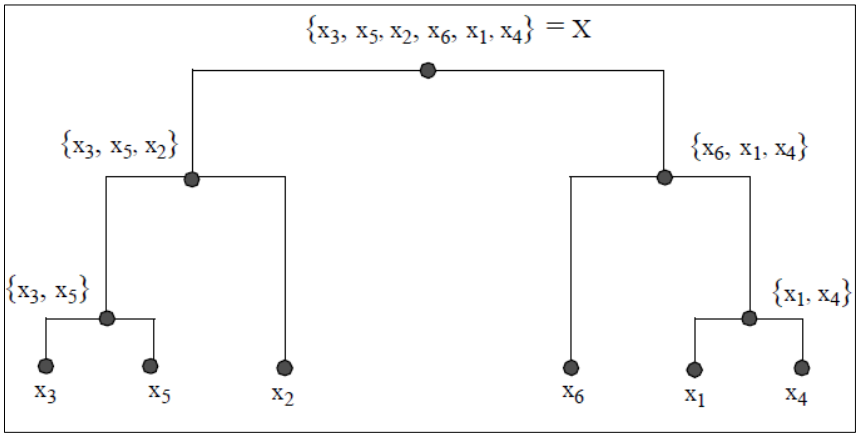


Figure 2.3: *Partitions emboîtées d'un ensemble X à 6 éléments [10].*

3.1.1 L'algorithme hiérarchique ascendant:

L'algorithme hiérarchique ascendant est décrit dans les étapes suivantes :

Au départ, on dispose de n objets à classer.

1. Associer chaque objet à classer à un nouveau cluster.
2. Calculer la matrice des distances qui sépare toutes les clusters deux à deux.
3. Chercher les deux clusters les plus proches.
4. Réunir les deux clusters les plus proches dans un seul nouveau cluster.
5. Mettre à jour la matrice des distances.
6. Réitère les étapes (3-5) jusqu'à ce que tous les clusters aient été regroupés.

Figure 2.4 : *L'algorithme hiérarchique ascendant [10].*

Selon la façon avec laquelle les clusters sont fusionnés, plusieurs algorithmes ont été réalisés.

Les algorithmes les plus utilisés dans la plupart des méthodes hiérarchiques ascendantes sont les algorithmes du lien simple ou saut minimum (single link), les algorithmes du lien ou diamètre complet ou maximal (complete link), les algorithmes du lien moyen (averagelink).

- Dans l'algorithme du lien simple, la distance entre deux clusters est la valeur minimum des distances entre toutes les paires d'objets, l'un du premier cluster, l'autre du deuxième.
- Dans l'algorithme du lien complet, la distance entre deux clusters est la valeur maximale des distances entre toutes les paires d'objets.
- Dans l'algorithme du lien moyen, la distance entre deux clusters est la valeur moyenne des distances entre toutes les paires d'objet, l'un du premier cluster, l'autre du deuxième [11].

3.1.2 L'algorithme hiérarchique descendant:

On procède selon les étapes suivantes:

Soit I l'ensemble des objets

1. Calculer les distances des éléments de I deux à deux et trier les valeurs par ordre décroissant.
2. Evaluer la distance max d(i_1, i_0) ou chaque élément de I est associé à un cluster C_0 représenté par i_0 ou C_1 représenter par i_1.
3. On considère le cluster C_i qui possède la distance maximale et on divise C_i en C_i^a et C_i^b puis on attribuant chaque élément de C_i à C_i^a ou C_i^b.
4. Recalculer les distances par ordre décroissant on s'arrête dès que le nombre des clusters égal la cardinalité de I.

Figure 2.5 : *L'algorithme hiérarchique descendant [11].*

3.2 L'algorithme Soustractive Clustering (SC):

Cette technique est appliquée lorsqu'il n'y a pas une idée claire sur le nombre des centres pour la répartition des données.

La méthode Soustractive Clustering est une extension de la méthode de classification proposée par Yager. Il suppose que chaque point de données est un centre potentiel de l'amas et calcule le potentiel de chaque point de données par la mesure de la densité des points de données l'entourant. [16]

L'algorithme sélectionne d'abord le point de données avec le plus grand potentiel en tant que centre de l'amas, puis évince les points de données à proximité du centre du premier groupe (déterminer par un rayon), puis calculer le centre prochain et ainsi de suite [17].

L'algorithme est fonctionné comme se suit:

1. Choisir le point de donnée avec le plus grand potentiel d'être le centre du premier groupe.
2. Évincer tous les points dans le voisinage du 1ercentre de l'amas (déterminé par le rayon).
3. Déterminer le prochain centre.
4. Itérer le processus jusqu'à ce que toutes les données soient dans un rayon du centre d'un amas.

Figure 2.6 : *L'algorithme Soustractive Clustering[17].*

Nous pourrons dire que l'algorithme SC réduit la complexité de calcul et donne une répartition des centres de cluster en fonction de la mesure de densité ainsi que du rayon (voir figure 3.6), sachant que chaque centre est un point de données avec le plus grand potentiel [17].

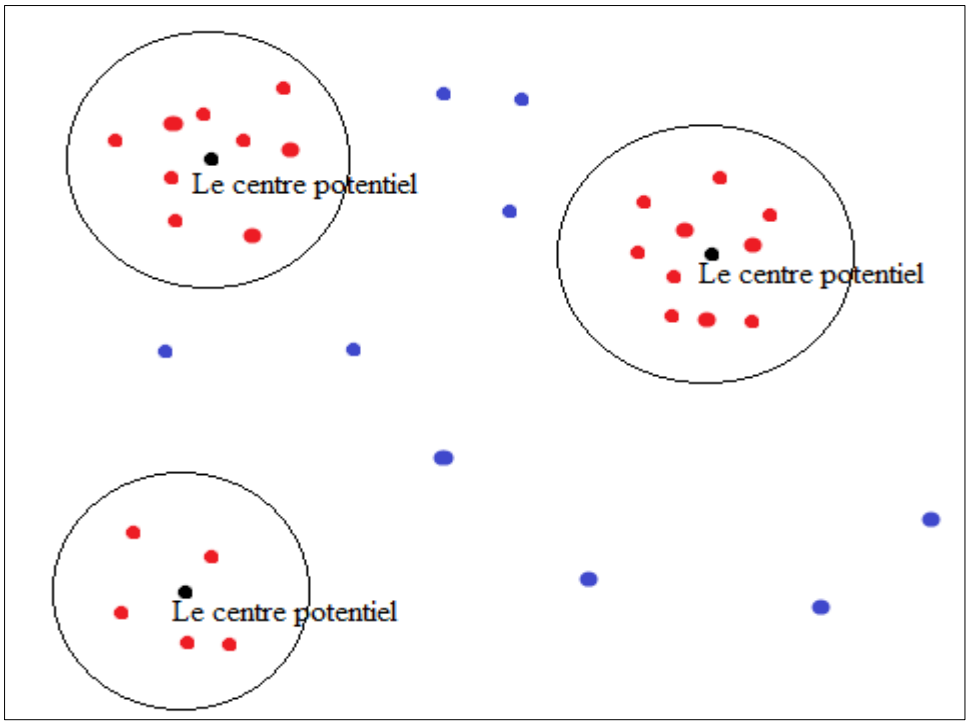


Figure 2.7 : *La distribution des données autour des centres d'amas dans l'algorithme Soustractive Clustering.[17]*

3.3 L'algorithme IsoData:

Dans la méthode K-means, le nombre de clusters K reste le même pendant toutes les itérations, cet inconvénient peut être surmonté dans l'algorithme IsoData (Itératif auto-organisation Analyse Technique Algorithme), qui permet de régler automatiquement le nombre des clusters lors de l'itération.

L'algorithme utilise également un certain nombre des paramètres pour déterminé les clusters à fusionner ou à diviser [w22]. Ces paramètres sont :

K : Le nombre initial des clusters.

I_{max}: Le nombre maximal des itérations.

σ_{max} : L'écart type maximum.

L_{min}: La distance minimale entre deux centres de clusters[w20].

On résume l'algorithmesuit les étapes suivantes :

1. Appliquer l'algorithme de k-means.
2. Si l'écart type d'un segment dépasse le maximum spécifié (σ_{max}), le segment est éclaté en deux segments.
3. Si la distance entre deux clusters (entre les centres des clusters) inferieur a L_{min} alors fusionner les deux clusters.
4. Aller à l'étape 1 jusqu'à I_{max}ou attendre quand les membres des groupes ne change plus.

Figure 2.8 : *L'algorithme IsoData[w23].*

Sachant qu'on peut utiliser un seul seuil tel que si la distance entre deux clusters inferieur à celle on doit la fusionner et dans le cas contraire on le segmenter.

Et voici l'algorithme k-means :

1. Choisir au hasard les centre de k clusters.
2. Attribuer chaque donner au centre le plus proche.
3. Recalculer les positions des nouveaux centroïdes.
4. Répéter les étapes 2 et 3 jusqu'à convergence, c'est-à-dire jusqu'à ce que les centroïdes ne bougent plus.

Figure 2.9 :*L'algorithme de K-means[w23].*

3.4 BDSCAN: (Density Based Spatial Clustering Applications With Noise):

3.4.1 Définitions :

BD SCAN (Applications de clustering spatial basées sur la densité avec bruit) reposent sur l'étude de la densité d'objets au voisinage de chaque objet.

L'algorithme de clustering DBSCAN repose sur une notion de cluster basée sur la densité et est conçu pour détecter les clusters de forme arbitraire ainsi que pour distinguer le bruit. [18]

3.4.2 Principe de BD SCAN :

DBSCAN permet l'identification des forme de cluster de forme arbitraire et le bruit dans une base de donné spatiale. Cet algorithme requiert seulement deux paramètres d'entrée afin que l'utilisateur puisse spécifier une valeur appropriée.

Eps(Epsilon) **:** Définit le voisinage autour d'un point de données

MinPts (Minimum points) : Nombre minimum de voisins (points de données) dans le rayon Eps

On fixe *Eps* le rayon de voisinage à étudier et *MinPts*, le nombre minimum de point qui doive être contenu dans le voisinage pour considérer la zone dense, l'idée clé de clustering basé sur la densité et que pour chaque point d'un cluster, ses voisins pour un rayon donné eps doit contenir un nombre minimum de point *MinPts*. Ainsi, le cardinal de son voisinage doit dépasser un certain seuil (considérer comme un objet principal). Ensuite DBSCAN collecte itérativement de proche en proche les objets atteignables par densité par rapport aux objets principaux, le processus se termine lorsqu'aucun nouveau point ne peut être ajouté à un cluster [w24]

Algorithmes

```
DBSCAN (DB, distFunc, eps, minPts) {
    C = 0 / * Compteur de cluster * /
    pour chaque point P dans la base de données DB {
        si le libellé (P) de non défini, continue / * Déjà traité en boucle interne * /
        Voisins N = RangeQuery (DB, distFunc, P, eps) / * Trouver des voisins * /
        si | N |   <minutes puis { / * Contrôle de densité * /
            étiquette (P) = bruit / * étiquette comme bruit * /
            continuer
        }
```

```
        C = C + 1 / * étiquette suivante du cluster * /
        étiquette (P) = C / * point initial de l'étiquette * /
        Ensemble de semences S = N \ {P} / * Voisins à développer * /
        pour chaque point Q dans S { / * Traite chaque point initial * /
            si étiquette (Q) = bruit, alors étiquette (Q) = C / * Modifier le bruit en point de
bordure * /
            si libellé (Q) ≠ non défini, continuer / * déjà traité * /
            label (Q) = C / * Label voisin * /
            Voisins N = RangeQuery (DB, distFunc, Q, eps) / * Trouver des voisins * /
            si | N |   Min Mints alors { / * Contrôle de densité * /
                S = S ∪ N / * Ajouter de nouveaux voisins au jeu de semences * /
            }
        }
    }
}
```

Figure 2.10 :*L'algorithme de DBSCAN [w24]*

3.4.3 Domaine d'utilisation :

Dans de grandes bases de données spatiales, Pour soutenir l'utilisation intensive du clustering dans la vision par ordinateur, reconnaissance, la récupération d'informations, l'exploration de données, etc. WEKA est un exemple de logiciel utilisant l'algorithme DBSCAN. quelques exemples d'application pratique de l'algorithme DBSCAN.

- Images satellites
- Cristallographie aux rayons x
- Détection d'anomalies dans les données de température [18]

3.4.4 Avantage et inconvénient :

L'algorithme DBSCAN est comparé à un autre algorithme de classification. Celui-là s'appelle CLARANS (Mise en cluster de grandes applications basées sur la recherche RAN domized). C'est une amélioration des algorithmes k-medoid (un objet du cluster situé près du centre du cluster, au lieu du point de gravité de la grappe, c'est-à-dire k-signifie). Les bonnes propriétés par rapport au k-medoide.

CLARANS fonctionne efficacement pour les bases de données contenant environ mille objets. Quand la base de données s'agrandit CLARANS prendra du retard car l'algorithme stocke temporairement tous les objets dans la mémoire principale, c'est-à-dire que le temps d'exécution augmentera [18]

3.4.4.1 Avantage :

- DBSCAN n'exige pas de spécifier le nombre de clusters dans les données à priori, par opposition à k-means .
- DBSCAN peut trouver des grappes formées de manière arbitraire. Il peut même trouver un cluster complètement entouré (mais non connecté) d'un autre cluster. En raison du paramètre MinPts, l'effet appelé lien unique (les différents clusters étant connectés par une fine ligne de points) est réduit.
- DBSCAN a une notification de bruit et est robuste aux valeurs aberrantes .
- DBSCAN nécessite seulement deux paramètres et est généralement insensible à l'ordre des points dans la base de données. (Cependant, les points situés à la limite de deux grappes différentes peuvent échanger leur appartenance à une grappe si leur régulation est modifiée et que l'affectation de grappe n'est unique que jusqu'à l'isomorphisme.)
- DBSCAN est conçu pour être utilisé avec des bases de données capables d'accélérer les requêtes de région, par exemple en utilisant une arborescence .
- Les paramètres MinPts et ε peuvent être définis par un expert de domaine, si les données sont bien comprises
- L'algorithme DBSCAN a bien réussi à classer tous les clusters. Néanmoins, c'est aussi supérieur lorsque l'on compare le temps d'exécution entre les deux algorithmes Le DBSCAN a une augmentation presque linéaire du temps de calcul, par rapport au nombre de points dans la base de données. Le gain CLARANS, cependant, est exponentiel et devient plus performant avec un facteur de 250 à 1900 dans cet essai. Le facteur continuera à grossir avec la taille de la base de données augmente.[18]

3.4.4.2 Inconvénient :

- DBSCAN n'est pas entièrement déterministe: les points frontières accessibles depuis plusieurs clusters peuvent faire partie de l'un ou l'autre des clusters, en fonction de l'ordre dans lequel les données sont traitées. Pour la plupart des ensembles de données et des domaines, cette situation ne se produit pas souvent et a peu d'impact sur le résultat de la mise en cluster à la fois sur les points centraux et les points parasites,
- DBSCAN est déterministe. DBSCAN est une variante qui traite les points de frontière comme du bruit, ce qui permet d'obtenir un résultat entièrement déterministe ainsi qu'une interprétation statistique plus cohérente des composants liés à la densité.
- La qualité de DBSCAN dépend de la mesure de distance utilisée dans la fonction region Query (P, ε). La métrique de distance la plus couramment utilisée est la

distance euclidienne . En particulier pour les données de grande dimension , cette métrique peut être rendue presque inutile en raison de la " malédiction de la dimensionnalité ", ce qui rend difficile la recherche d'une valeur appropriée pour ε. Cet effet, cependant, est également présent dans tout autre algorithme basé sur la distance euclidienne.

- DBSCAN ne peut pas bien regrouper des ensembles de données avec de grandes différences de densités, car la combinaison minPts-ε ne peut pas être choisie de manière appropriée pour toutes les grappes.
- Si les données et l'échelle ne sont pas bien comprises, il peut être difficile de choisir un seuil de distance significatif ε. [18]

4 Conclusion:

La classification non supervisée, ou clustering, est un thème de recherche majeur en apprentissage automatique, en analyse et en fouille de données ainsi qu'en reconnaissance de formes. Il fait partie intégrante de tout un processus d'analyse exploratoire de données permettant de produire des outils de synthétisation, de prédiction, de visualisation et d'interprétation d'un ensemble d'individus (personnes, objets, processus, etc.).

Dans le chapitre suivant nous donnerons les détails de conception de notre solution pour l'identification des services à partir d'une application orientée objet en utilisant les algorithmes de classification, tel que DBSCAN.

Partie II

Chapitre3

Conception

1 Introduction:

Vu les avantages apportés par l'architecture orientée service dans le domaine des applications distribuées, plusieurs chercheurs se sont concentrés sur les stratégies et méthodes de migration vers cette dernière. Plusieurs approches existent, mais toutes s'accordent sur une étape commune qui est l'identification des services. L'identification peut se faire de façon manuelle, semi-automatique ou automatique. Dans notre cas nous avons choisi de d'automatiser cette étape. Pour le faire nous avons choisi d'utiliser les méthodes de clustering pour regrouper automatiquement les classes en services.

Le clustering est une méthode non supervisée visant à organiser un ensemble d'objets en groupes ou clusters, de façon à avoir des objets similaires groupés et les objets différents organisés dans des groupes différents. Dans notre cas nous avons choisi d'adapter l'algorithme DBSCAN pour modéliser notre problème et automatiser l'identification.

2 L'objectif global :

L'objectif de notre travail s'inscrit dans le cadre de la réingénierie des systèmes orientés objet vers des systèmes orientés services. Nous nous sommes concentrés sur l'étape d'identification des services puisque c'est une étape primordiale dans tout processus de réingénierie vers une architecture orientée service.

Puisqu'une application orientée objet est constituée d'un ensemble d'objets et une application orientée service est constitué d'un ensemble de services de granularité plus gros et qui offre plus de fonctionnalités, nous avons pensé à modéliser le problème d'identification de service à un problème de regroupement de classes dans un module plus grand appelé service qui doit respecter les caractéristiques de service. Ainsi les classes regroupées doivent être fortement couplées et fortement cohésives pouvant constituer un module autonome, réutilisable et distribuable. Donc un groupe des classes qui respecte la définition d'un service de qualité.

Donc pour synthétiser, le but est d'utiliser les algorithmes de regroupement pour automatiser l'identification des services à partir d'applications orientée objet.

Pour atteindre ce but nous devions choisir un algorithme de regroupement et l'adapter à notre problématique pour nous permettre d'identifier des services qu'on appellera services candidats à partir du code source d'applications orientée objet.

Un travail similaire a déjà été initié en2015 [21]. Nous voulons l'étendre de tester de nouveaux algorithmes et comparer les résultats.

3 Choix des algorithmes de regroupement

Pour choisir un algorithme de regroupement, nous avons dû explorer un nombre important d'algorithmes de regroupement existants.

3.1 Les algorithmes à base du Deep Learning

Nous avons essayé dans un premier temps d'adapter les algorithmes du Deep Learning à notre problématique, malheureusement ces algorithmes sont à base de réseaux de neurone, et nécessitent d'une part une étape d'apprentissage et d'autre part imposent que le nombre de classe sur lesquelles vont être distribués les éléments doit être connue. Dans notre cas le nombre de services résultant du regroupement ne peut être connue à l'avance. Donc ce type d'algorithme ne pouvait pas être adapté à notre problématique.

Une autre tentative d'utilisation des cartes auto-organisateur de Kohonen, malheureusement, cette tentative n'a pas abouti aussi.

3.2 Les algorithmes de regroupement non supervisés

Puisque le nombre de services résultant ne peut être connu à l'avance, nous nous sommes penchés sur l'étude des algorithmes de regroupement non supervisé. Nous avons exploré un nombre important d'algorithmes que nous avons détaillé dans le chapitre précèdent. Notre choix s'est porté enfin sur l'algorithme DBSCAN avec des adaptations qui nous permettent d'identifier les services candidats.

4 Processus d'identification des services :

Pour identifier les services candidats, nous avons suivi les mêmes étapes suivies par l'étude [21] et nous avons récupéré la fonction objective qui calcule la distance entre les classes.

Pour automatiser la transformation des applications orientées objet vers des applications orientées service, nous avons suivi les étapes suivantes :

- **La mise en correspondance Objet-Service :** c'est de recenser les caractéristiques des éléments de base des deux formalismes qui sont la classe (ou objet) et le service afin de faire une correspondance entre les deux.
- **Analyse du code source:** analysé le code source d'une application O.O afin d'extraire toutes les expressions les appels des méthodes, les utilisations des attributs…etc., afin de récupérer les caractéristiques à travers lesquelles nous identifierons les services.

- **Identification des services :** décomposé une application en un ensemble de services, en regroupant les classes similaires à l'aide des algorithmes dans ce cas on va utiliser DB SCAN comme algorithme.

Pour passer de la phase d'analyse à la phase d'identification nous avons suiviles étapes suivantes :

- ✓ **Calcul des métriques :** Il s'agit de quantifier et de mesurer les propriétés des services, dans notre cas le couplage et la cohésion entres les classes candidates pour former un service.
- ✓ **Définition de la fonction objective :** Combiner les métriques proposées pour avoir une fonction globale qui mesure les propriétés d'un service.
- ✓ **La matrice symétrique :** C'est le résultat de l'application de la fonction objectif sur les différentes classes de l'application orientée objet.

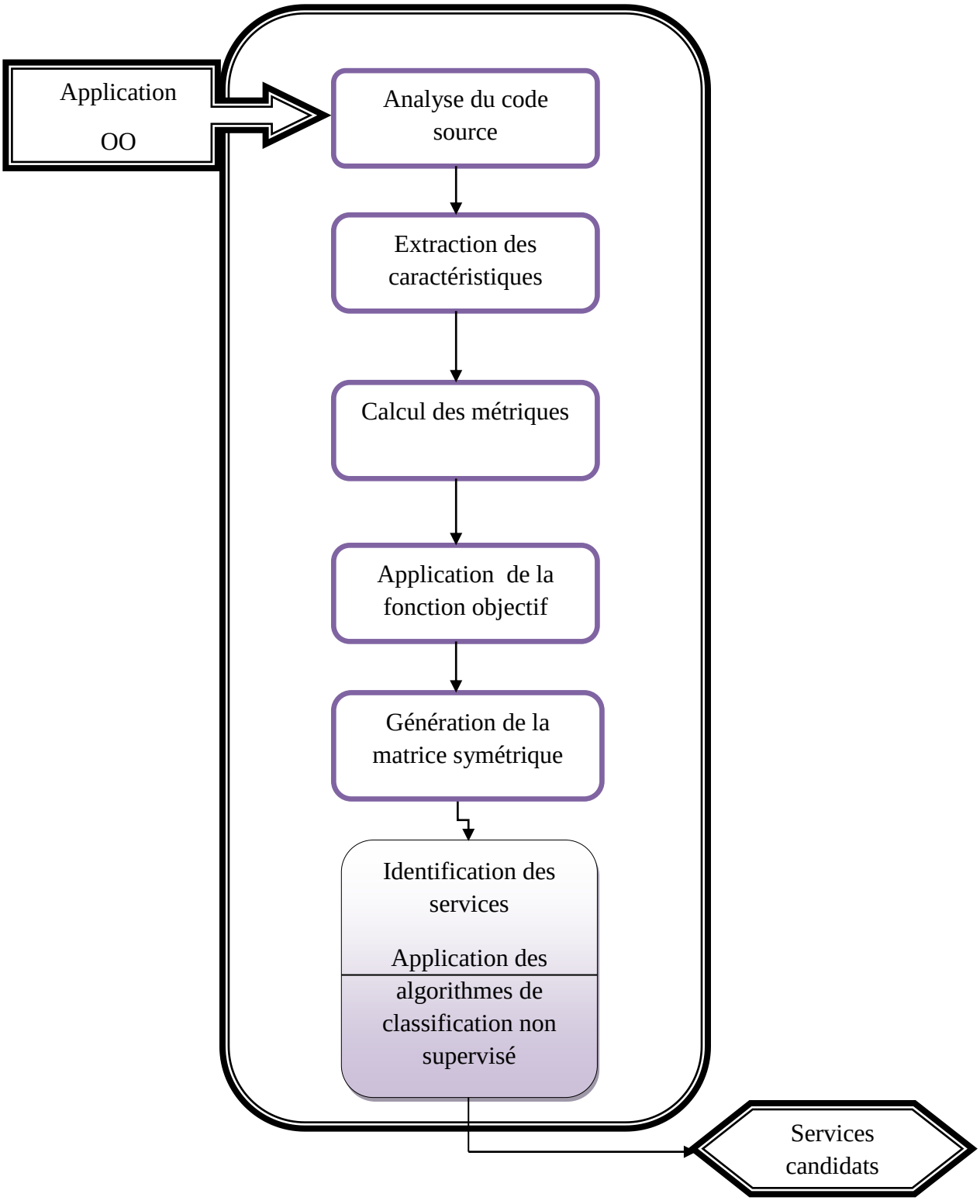


Figure 3.1 : *Processus général d'identification des services.[21]*

4.1 La mise en correspondance Objet-Service:

Considérons un service comme un groupe de classes définies dans l'orienté objet. Parmi ces classes il y'a celles qui définissent des opérations (classes interface).

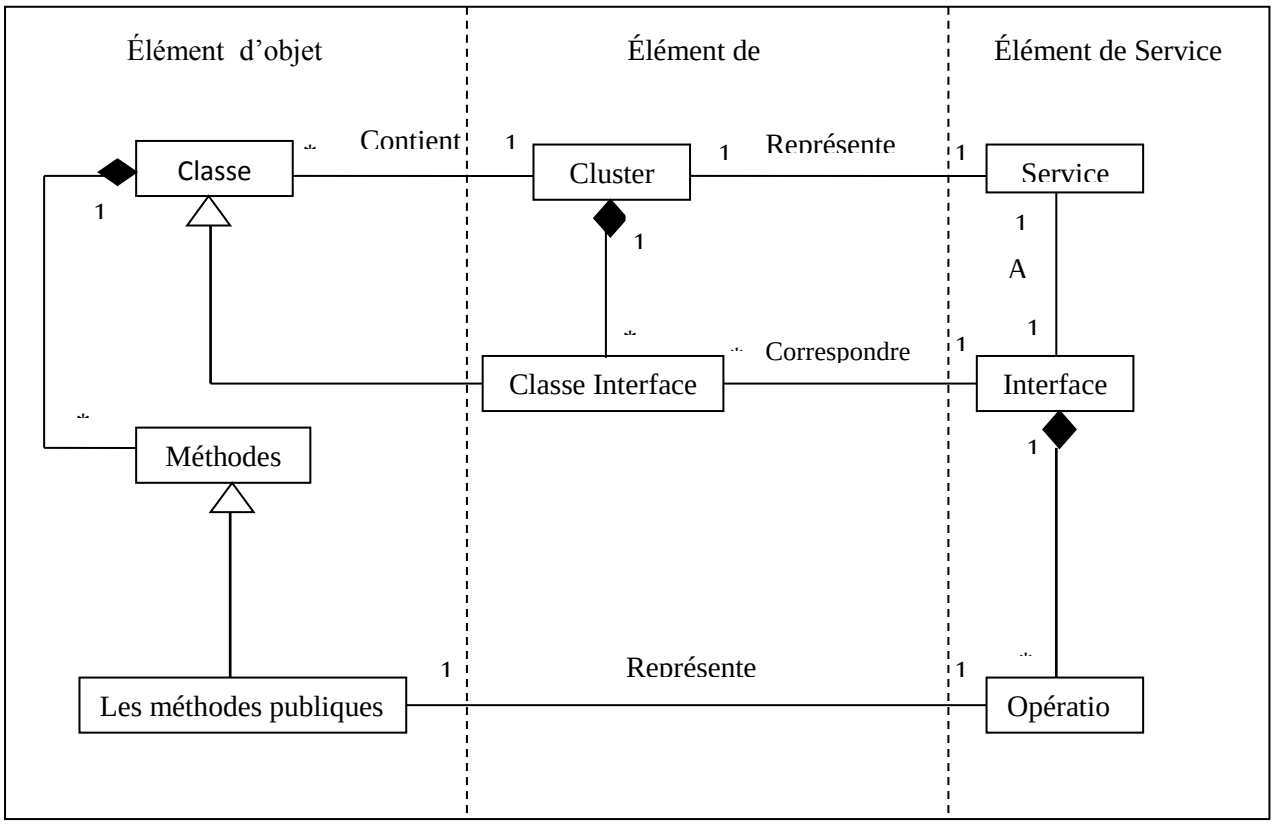


Figure 3.2: *La mise en correspondance entre l'orienté objet et l'orienté service [31].*

4.2L'analyse du code source:

L'étape d'analyse permet d'extraire les informations d'une application donnée. Ces informations concernant les classes qui composent un projet, et qui nécessite un parcours de la totalité du projet pour atteindre ces derniers.

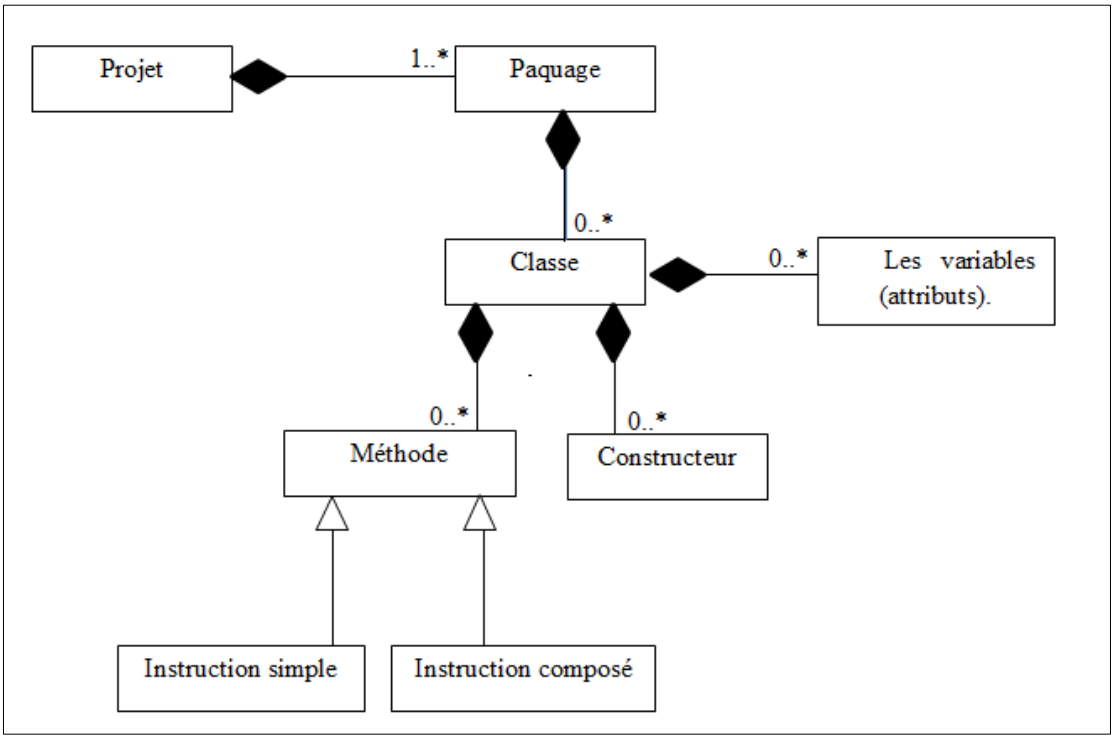


Figure 3.3 : *La structure des applications orientées objet.*

4.3 Le calcul des métriques :

Dans cette partie on base sur l'étude des éléments de base de notre application (les classes) afin de chercher les métriques qui mesurent la similarité entre ces éléments.

Afin de mesurer ces caractéristiques, nous avons trouvé que les métriques de couplage et de cohésion conviennent bien à nos besoins.

4.3.1 Mesure de couplage :

C'est le degré d'interdépendance entre modules. Cette notion est très utilisée dans le génie logiciel et en particulier dans le paradigme objet [22].

Il existe des nombreuses formules qui permettant de mesurer le couplage des classes avec l'extérieur [23, 24, 25, 26, 27, 28]. Une classe est couplée avec une autre classe si elle est en relation avec une ou plusieurs parties de code source de celle-ci.

Voici quelques mesures de couplage :

- NIH_ICP[27]: mesure le nombre d'appels de méthodes des classes avec lesquelles une classe n'a pas de relation d'héritage.
- DAC[28]: mesure le nombre d'attributs de la classe qui ont pour type une autre classe.

- OCMIC[23]: mesure le nombre des paramètres qui ne sont pas de type aou primitifs dans les méthodes de la classe a.

Nous avons combiné ces trois formules dans la formule suivante, étant donné que le couplage est une relation binaire entre deux classes(A, B) :

Couplage (A, B) = NIH_ICP (A, B) + DAC (A, B) + OCMIC (A, B).

4.3.2 Mesure de cohésion:

La cohésion mesure la force de la collaboration au sein d'un ensemble d'éléments. Elle doit mesurer la cohésion d'un ensemble de classes ainsi que la cohésion d'une classe [22].

La métrique de cohésion prend en considération les relations directes et les relations indirectes.

- **Cohésion basée sur la relation directe:**

Deux méthodes Mi et Mj peuvent être directement connectées de différentes manières: Elles partagent au moins une variable d'instance en commun (relation UA : usage d'attributs), ou interagissent au moins avec une méthode de la même classe (relation IM invocation de méthodes). Donc, deux méthodes peuvent être directement connectées par un ou plusieurs critères.[29]

$UA_{Mi} \cap UA_{Mi} \neq \emptyset$ ou $IM_{Mi} \cap IM_{Mj} \neq \emptyset$.[29]

Considérons un graphe non dirigé G_D où chaque nœud représente une méthode de la classe. Il y'a un arc entre deux méthodes Mi et Mj si elles sont directement reliées. Soit E_D le nombre d'arcs dans le graphe G_D.

Le degré de cohésion dans la classe C basé sur la relation directe entre ses méthodes est défini par:

$DC_D = |E_D| / [n*(n-1)/2]$.[29]

Tel que DC_D : donne le pourcentage de paires de méthodes publiques qui sont directement reliées, et n c'est le nombre des méthodes [29].

- **Cohésion basée sur la relation indirecte:**

Deux méthodes Mi et Mj peuvent être indirectement reliées si elles sont directement ou indirectement reliées à une méthode Mk.

Une méthode est indirectement reliée par une autre méthode s'il existe une séquence de méthodes M1, M2,…Mk telle que Mi est directement connectée à Mi+1 (i=1, k-1).

Considérons maintenant un graphe non orienté G_I où les sommets sont les méthodes publiques d'une classe C. Il y'a un arc entre deux sommets si les méthodes correspondantes sont directement ou indirectement reliées.

Soit E_I, le nombre d'arcs du graphe G_I, Alors, le degré de cohésion dans la classe C est défini par:

$$DC_I = |E_I| / [n*(n - 1) / 2] \text{ [29].}$$

DC_I : Donne le pourcentage de paires de méthodes publiques qui sont directement ou indirectement reliées [29].

Donc la cohésion d'une classe C est donnée par la formule suivante :

$$DC=DC_D+DC_I => DC=|E_D|+|E_I| / [n*(n - 1) / 2] \text{ [29].}$$

Remarque:

$$\text{Cohésion (A, B) = Cohésion (B, A).[29].}$$

$$0 \leq \text{Cohésion } (A_i, A_j) \leq 1.$$

Le calcul du couplage et de la cohésion peut être résumé par la figure 3.4.

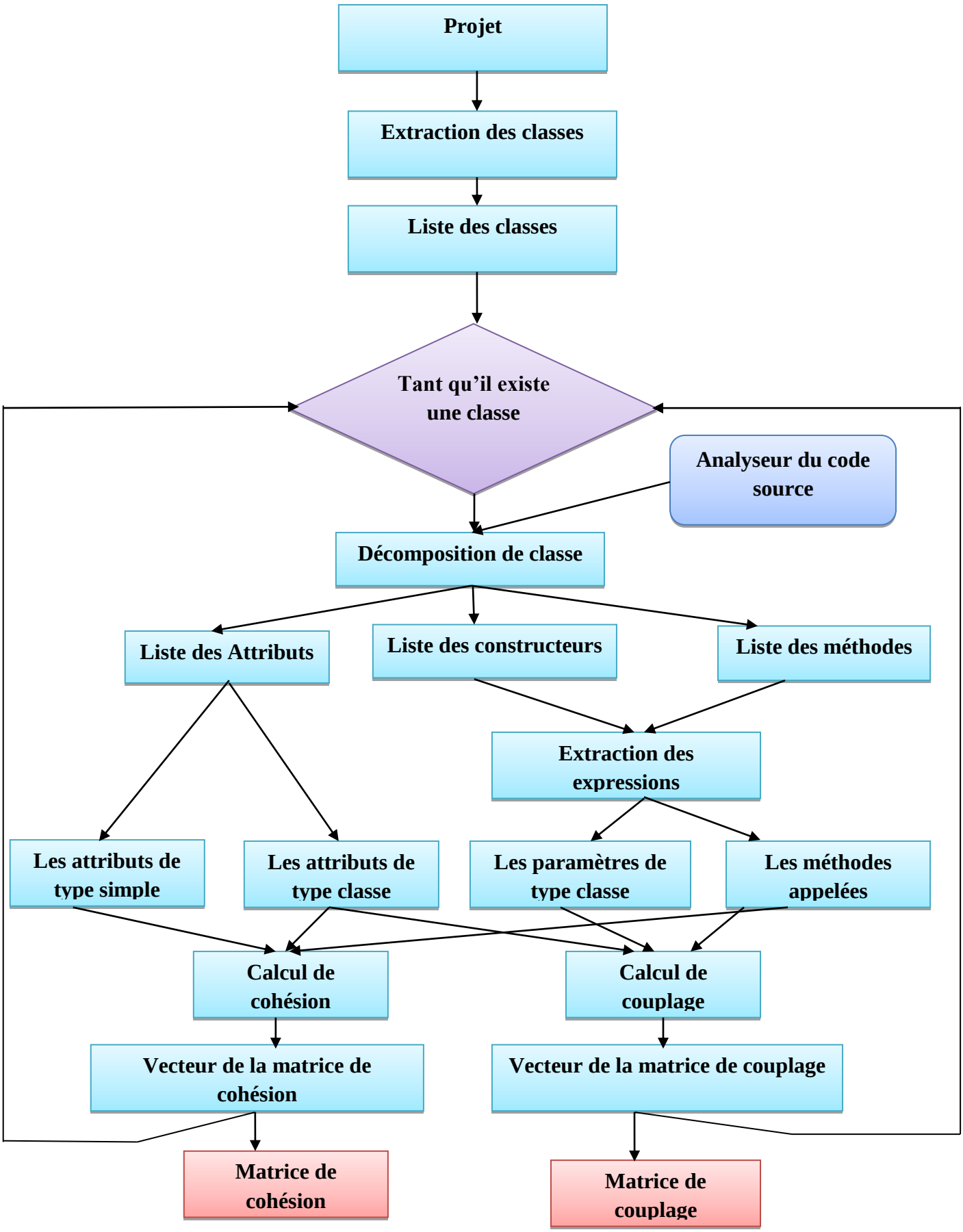


Figure 3.4: *Cycle de calcul de couplage et cohésion d'une application OO [21]*

4.4 La spécification de la fonction objective:[21]

La fonction objective est une fonction qui permet de calculer le degré de proximité entre deux classes. Dans notre cas elle remplace la fonction qui calcule la distance entre deux points dans les algorithmes de regroupement.

Cette fonction est calculée à partir du calcul du couplage et de la cohésion, nous l'avons récupéré directement de [21].

La formule finale de la fonction objective est la suivante :

$$F(A,B) = \frac{NB\_FCT\ (A,B) + (1 - Cohésion\ (A,B)) + (1 - MoyenneCouplage(A,B))}{3}$$

A,B : deux classes.

NB_FCT: Le nombre des fonctionnalités public n'étant pas appelé à l'intérieur du service (représente l'interface de service)/ nombre total des méthodes.

Cohésion(S) : La cohésion intra service (entre les classes du service).

Couplage(S) : Le couplage intra service.

Les valeurs de la fonction objective entre 0 et 1.

Le résultat de la fonction objectif représente le degré de similarité entre deux classes.

4.4.1 Génération de la Matrice des similarités :

L'application de cette formule sur les classes deux a deux génère une matrice symétrique où la plus petite valeur signifie que le degré de similarité entre les classes est le plus élevé.

4.5 L'identification des services :

Cette phase c'est la phase la plus importante dans notre travail, elle porte sur l'adaptation de l'algorithme DBSCAN afin d'identifier les services candidats d'une application.

4.5.1Principe d'algorithme DBSCAN

L'algorithme DBSCAN nécessite deux paramètres -

Eps: définit le voisinage autour d'un point de données, c'est-à-dire que si la distance entre deux points est inférieure ou égale à «eps», ils sont considérés comme voisins. Si la valeur eps est choisie trop petite, une grande partie des données seront considérées comme des valeurs aberrantes. S'il est choisi très grand, les clusters vont fusionner et la majorité des points de

données seront dans les mêmes clusters. Une façon de trouver la valeur eps est basée sur le graphique de k-distance. [18]

MinPts: Nombre minimum de voisins (points de données) dans le rayon eps. Plus le jeu de données est grand, plus la valeur de MinPts doit être grande. En règle générale, le nombre minimal de MinPts peut être dérivé du nombre de dimensions D dans le jeu de données, sous la forme MinPts> = D + 1. La valeur minimale de MinPts doit être choisie au moins 3. [18]

Dans cet algorithme, nous avons 3 types de points de données. [18]

- **Core Point (Point central):** Un point est un point central s'il a plus de points MinPts dans l'eps.
- **Border Point :** Un point qui a moins de MinPts au sein de l'eps mais qui se trouve au voisinage d'un point central.
- (**Noise or outlier**) Bruit ou valeur aberrante : Point qui n'est pas un point central ou un point frontière.

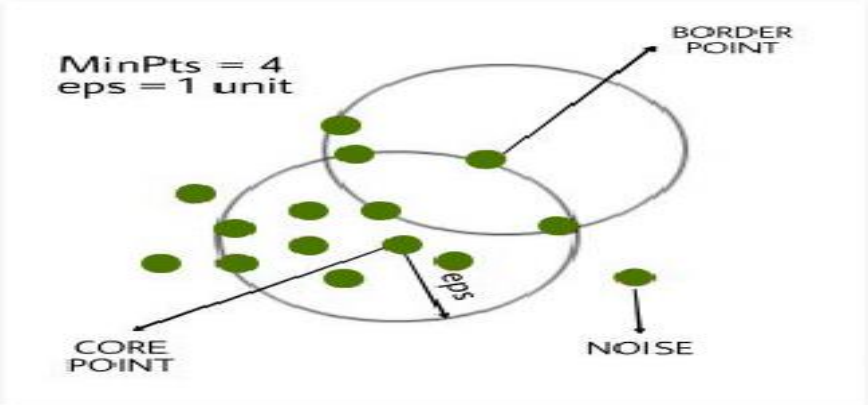


Figure 3.6 : *exemple sur algorithme DBSCAN.*

- DBSCAN (Density Based Spatial Clustering of Application with Noise), est introduit par Ester, Kriegel, Sander et Xu. [19] voici l'algorithme DBSCAN de la manière suivante

1 Sélectionner aléatoirement un objet p de l'ensemble d'objets X.

2 Déterminer l'ensemble E des objets accessible depuis p dans X.

3 Si p est un noyau, alors E est une classe.

4 Sinon sélectionner un autre objet et aller en (2).

5 S'arrêter lorsque tous les objets ont été sélectionnés.

Figure 3.7 : *Principe l'algorithme DBSCAN [19]*

- Les deux paramètres nécessaires de l'algorithme DBSCAN sont :

— Le rayon maximum situant les voisins : Epsilon

— Le nombre minimum de points dans le voisinage-Epsilon d'un point : min Points

Pour former un cluster, on doit avoir un nombre de points (ou des instances) situés dans un rayon inférieur à Epsilon qui est supérieur ou égal au nombre minimum de points (min Points). Un Point P est dense-accessible à partir de Q (c'est-à- dire Q est le centre de cluster) seulement si : 1. La distance entre P et Q (d(P,Q)) est inférieure ou égale au rayon maximum situant les voisins (Epsilon) c'est-à-dire : d(P,Q) <= Epsilon et P est au voisinage de Q. 2. Le nombre de points au voisinage de Q est supérieur ou égal au nombre minimum de points, c'est à-dire | Q/ d(P,Q) <= Epsilon | >= min Points[20].

DBSCAN sous forme d'algorithmes [w25]

```
1. Prendre un point x qui n'a pas été visité

2. Construire N = voisinage(eps, x)

3.  If |N| <n_min:

        Marquer x comme bruit

Else:

Initialiser C = {x}
Agrandir_cluster(C, N, eps, n_min)

    Ajouter C à la liste des clusters
    Marquer tous les points de C comme visités

4. Repeat 1-3 until tous les points ont été visités



agrandir_cluster(C, N, eps, n_min):
    For u in N:
If u n'a pas été visité:
N' = N(eps, u)
        If |N'| >= n_min:
N = union (N, N')
        If u n'appartient à aucun autre cluster:
            Ajouter u à C
```

Figure 3.8: *l'algorithme DBSCAN [w25].*

4.5.2 L'adaptation de l'algorithme DBSCAN

Pour DBSCAN, les paramètres Eps et MinPts sont nécessaires. Les paramètres doivent être spécifiés par l'utilisateur. Idéalement, la valeur d'Eps est donnée par le problème à résoudre (par exemple, une distance physique), et MinPts est alors la taille de cluster minimale souhaitée

Nous avons adapté cet algorithme de la façon suivante :

4.5.3 Détermination des paramètres Eps et MinPts

- ***Eps*** *:*les distances entre les classes sont calculées à partir de la fonction objective, détaillée à priori ; toute les distance sont entre 0 et 1. Choisir un Eps supérieur a 0.5 cela implique que le résultat sera un seul cluster donc un seul service sera identifié. Nous avons choisi des valeurs initialement inférieures à 0.3. La valeur optimale sera choisie de façon empirique après plusieurs exécutions de l'algorithmes et l'analyse des résultats.

- ***MinPts :*** Ce paramètre qui définit le nombre de points minimum dans un cluster est généralement fixé au moins à 3, dans notre cas nous acceptons cette valeur telle quelle est en proposant à l'utilisateur de la fixer autrement surtout dans le cas où le nombre de classe-objet devient de plus en plus grand.

- ***Les points aberrants*** : Ce sont les points qui ne sont pas affecté à aucun cluster, dans notre cas ; nous n'allons pas les ignorer ; mais nous allons les déclarer comme service a grain fin constitué de 1 ou 2 classe-objet.

- ***Calcul de voisinage*** : pour vérifier la distance entre les classes-objet on applique une fonction qui va nous fournir une valeur entre 0 et 1 qui représente le degré de proximité.

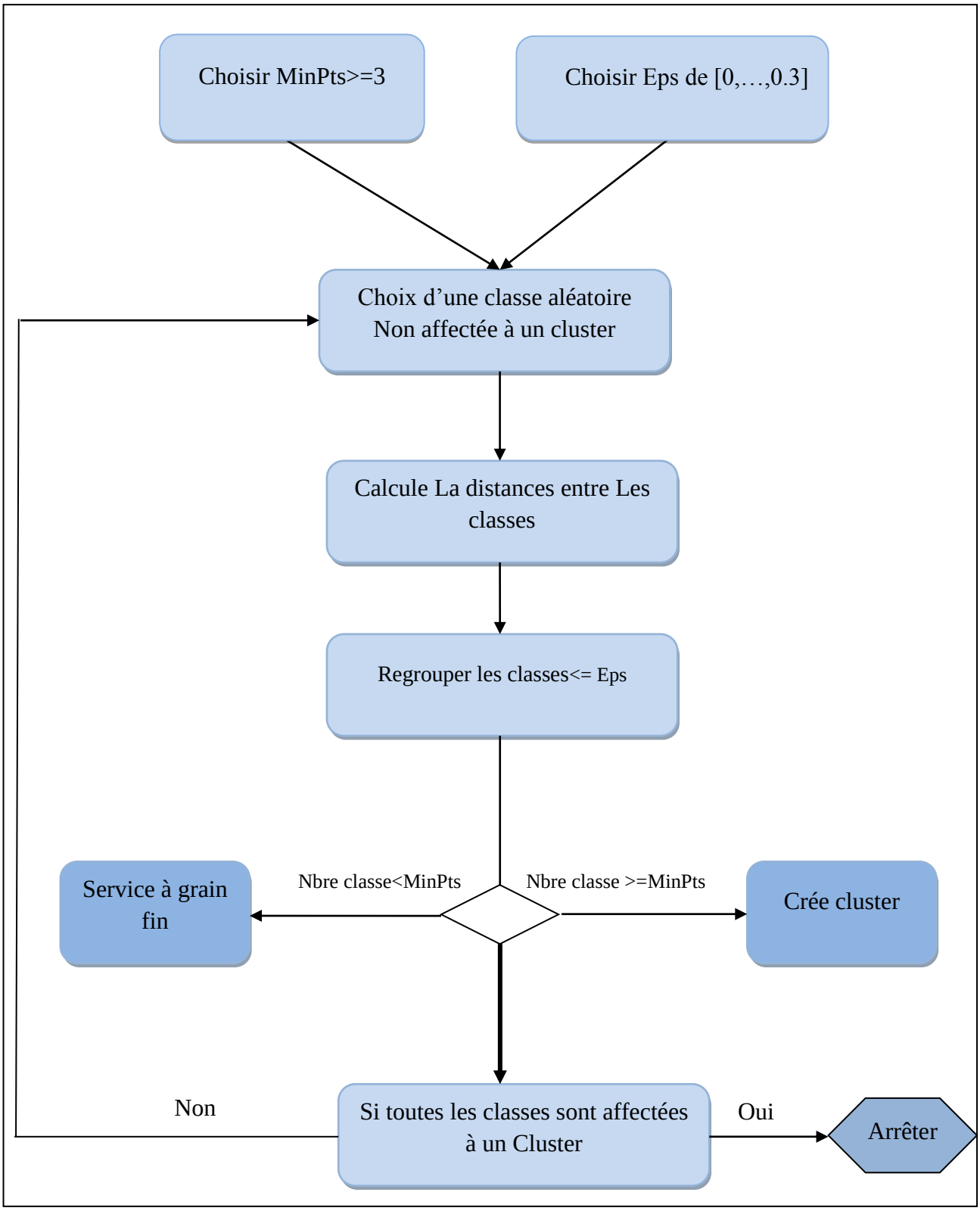


Figure 3.9: *Diagramme du déroulement de l'algorithme DBSCAN Adapté*

6 Conclusion :

La conception consiste à présenter la méthode suivie pour résoudre la problématique décrite, Cette phase est très importante pour le développement du logiciel.

Dans ce chapitre nous avons détaillé les différentes étapes de notre projet à partir de l'analyse du code source, puis la décomposition de l'application jusqu'à l'identification des services candidats.

Pour former des groupes de classes qui peuvent être considérés comme des services candidats, nous avons dû faire une correspondance entre les deux paradigmes objet et service puis extraire les caractéristiques d'un service de qualité.

L'identification des services étant modélisée sous forme de problème de regroupement de classe. Nous avons choisi d'adapter l'algorithme de classification non supervisée (DBSCAN) puisque le nombre de services résultant ne peut être connu à l'avance.

Les résultats de l'étude comparative seront détaillés dans le chapitre suivant.

Chapitre4

Implémentation

1 Introduction :

Dans cette section nous allons présenter les détails d'implémentation de notre projet, l'objectif global étant d'automatiser l'identification des services à partir d'une application orientée objet, ce qui a orienté notre choix pour le langage JAVA avec l'éditeur Eclipse.

Tout d'abord nous commençons par présenter les outils de développement que nous avons utilisé. Puis, nous présenterons les interfaces et le déroulement de l'application développée.

2 Les outils de développement :

2.1 Java :

Nous avons choisi Java comme langage de programmation parce que c'est un bon représentant des langages orientées objet assez puissant et fiable avec l'éditeur Eclipse, qui offre des interfaces faciles à manipuler.

2.2WEKA: Waikato Environment for Knowledge Analysis

En français : « environnement Waikato pour l'analyse de connaissances » c'est une suite de logiciels d'apprentissage automatique écrite en Java et développée à l'université de Waikato en Nouvelle-Zélande. Weka est un logiciel libre disponible sous la Licence publique générale GNU (GPL). [w26].

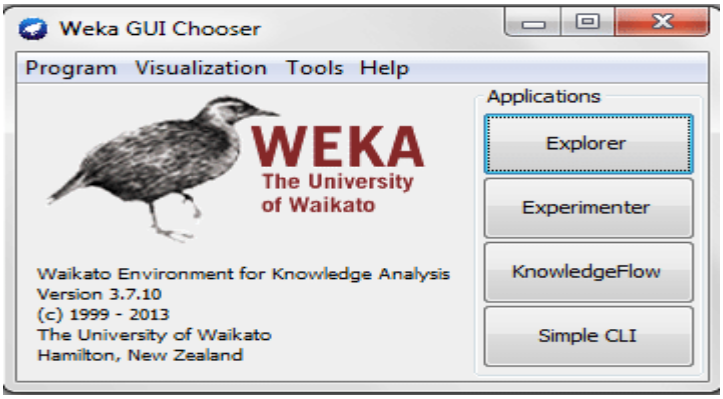


Figure 4.1: *Les Composant graphique de WEKA.*

2.3ASTParser :

ASTParser est un analyseur syntaxique qui convertit un fichier source java à un arbre syntaxique abstrait AST.

L'analyseur syntaxique (ASTParser) permet d'analyser et de découper les différents éléments du langage Java. Il permet ainsi d'extraire les différentes composantes d'une classe donnée (le nom des classes, les importations, les méthodes, les appels, les variables, etc).

ASTParser est disponible sous forme d'une bibliothèque (JAR) (sur le site https://code.google.com/p/javaparser/) utilisable dans beaucoup d'environnements. Cette bibliothèque est un projet écrit entièrement en Java et ne requiert aucun environnement particulier. Il est indépendant de l'IDE ou de toute autre technologie, il est simple à utiliser et de petite taille [w27].

Nous l'avons utilisé pour parcourir tous les éléments d'une application java et extraire les relations qui nous permettent de calculer la fonction objective basée sur le couplage, la cohésion et l'héritage.

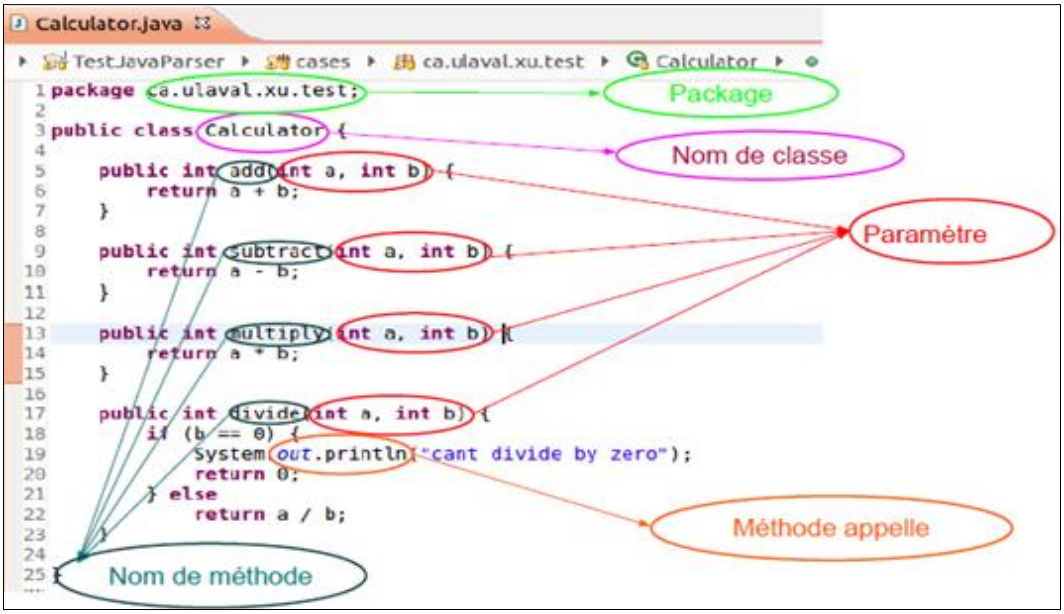


Figure 4.2: *Les composantes d'une classe java.*

Voici un exemple de quelques éléments en java et les éléments qui correspondant en ASTParser:

L'élément en java.	L'élément correspond en ASTPareser.
Package	PackageDeclaration
Import	ImportDeclaration
JavaClass	ClassOrInterfaceDeclaration

Method	MethodDeclaration,ConstructorDeclaration
Field	FieldDeclaration
MethodCall	MethodCallExpr
Parameter	Parameter
Variable	VariableDeclaratorId

Tableau 4.1 : *Quelques éléments en java et leur correspondance en ASTParser [w27].*

3 L'architecture globale de notre application:

L'utilisateur charge une application orientée objet à partir de l'interface, cette dernière est analysée par l'analyseur ASTParser. Les informations des différentes parties de l'application seront extraites puis envoyées au package de couplage et de cohésion pour calculer ces deux métriques. Pendant le calcul de la cohésion, un calcul parallèle est effectué pour récupérer le nombre des fonctionnalités, ensuite nous appliquons la fonction objective afin de générer la matrice de similarité qui constitue l'entrée du package d'identification de service. Le package d'identification permet d'appliquer les algorithmes de classification pour proposer à la fin à l'utilisateur des services candidats.

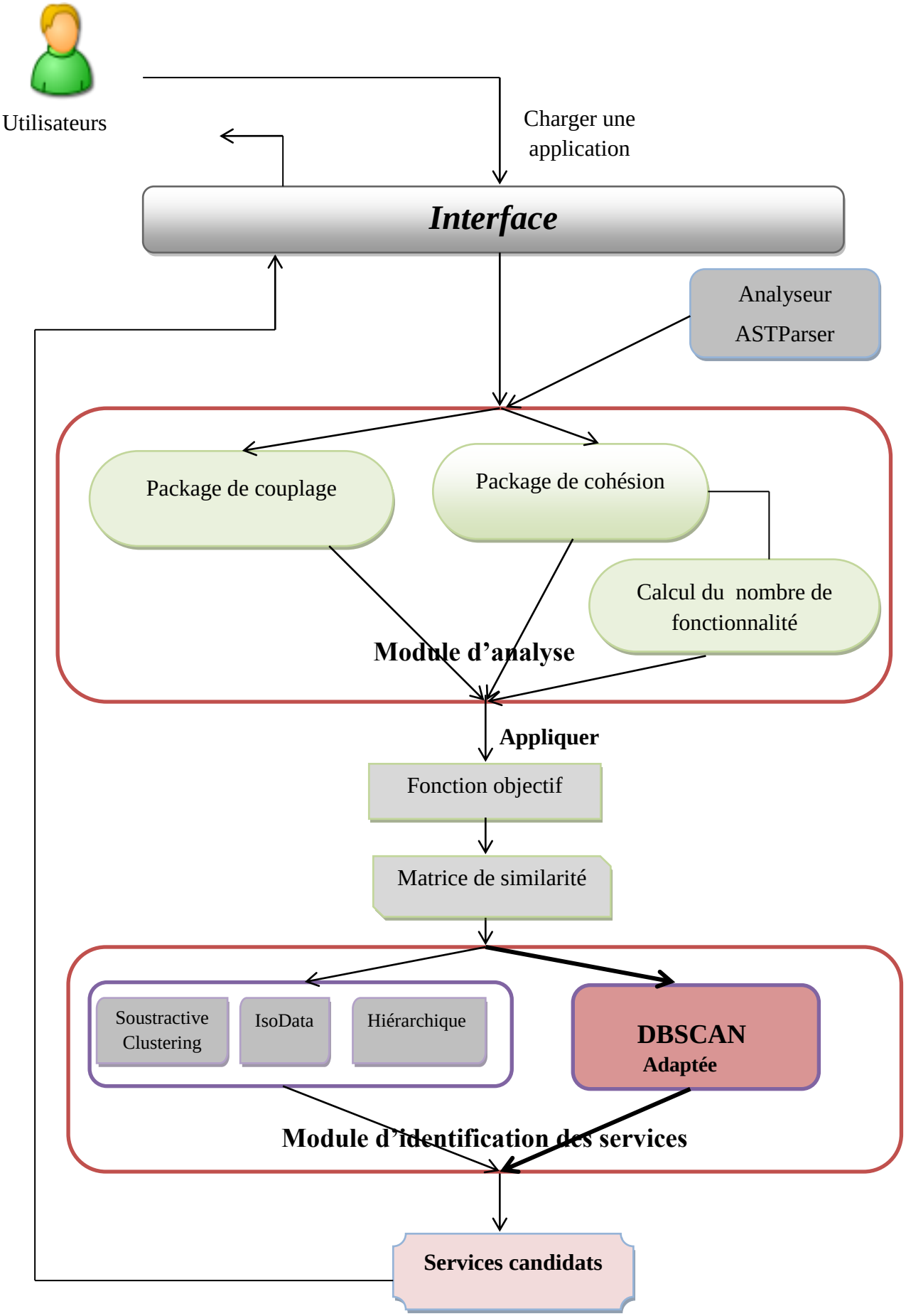


Figure 4.3 : *Architecture global de notre application.*

4 L'interface de l'application :

Notre interface est divisée en deux parties : une contenant les options d'accueil et l'autre réservéeà l'identification des services qui aussi composé en deux partie (voir figure 4.4) :

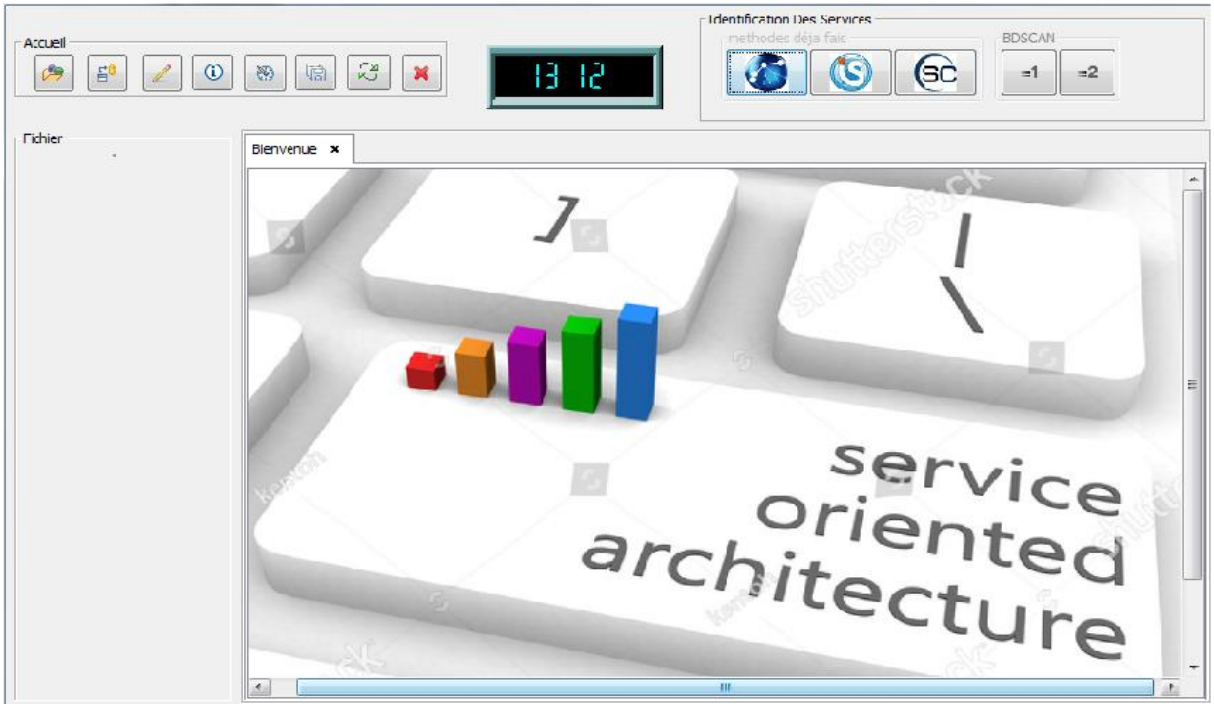


Figure 4.4 : *L'interface de notre application.*

4.1 Options d'accueil :

Contient les fonctionnalités suivantes :

- ✓  Pour charger un projet java.
- ✓  Récupérer une matrice de similarité à partir d'un fichier texte.
- ✓  Remplir une matrice de similarité.
- ✓  Guide d'utilisation.
- ✓  Calculer le temps d'exécution.
- ✓  Sauvegarder une matrice.
- ✓  Restaurer les fenêtres précédentes.

✓  Quitter l'application.

La partie suivante détaille chaque fonctionnalité.

4.1.1 Charger un projet java : 

Une fois que nous cliquons sur ce bouton une fenêtre s'affiche permettant aux utilisateurs de choisir et d'importer un projet java, après le choix du projet une fenêtre qui indique le nombre total des classes apparait puis l'analyseur commencera à filtrer tous les fichiers source et de calculer le couplage, la cohésion, le nombre des fonctionnalités ainsi que le calcul de la matrice de similarité (figure 4.5).Puis toutes les informations seront affichées dans des fenêtres séparées.

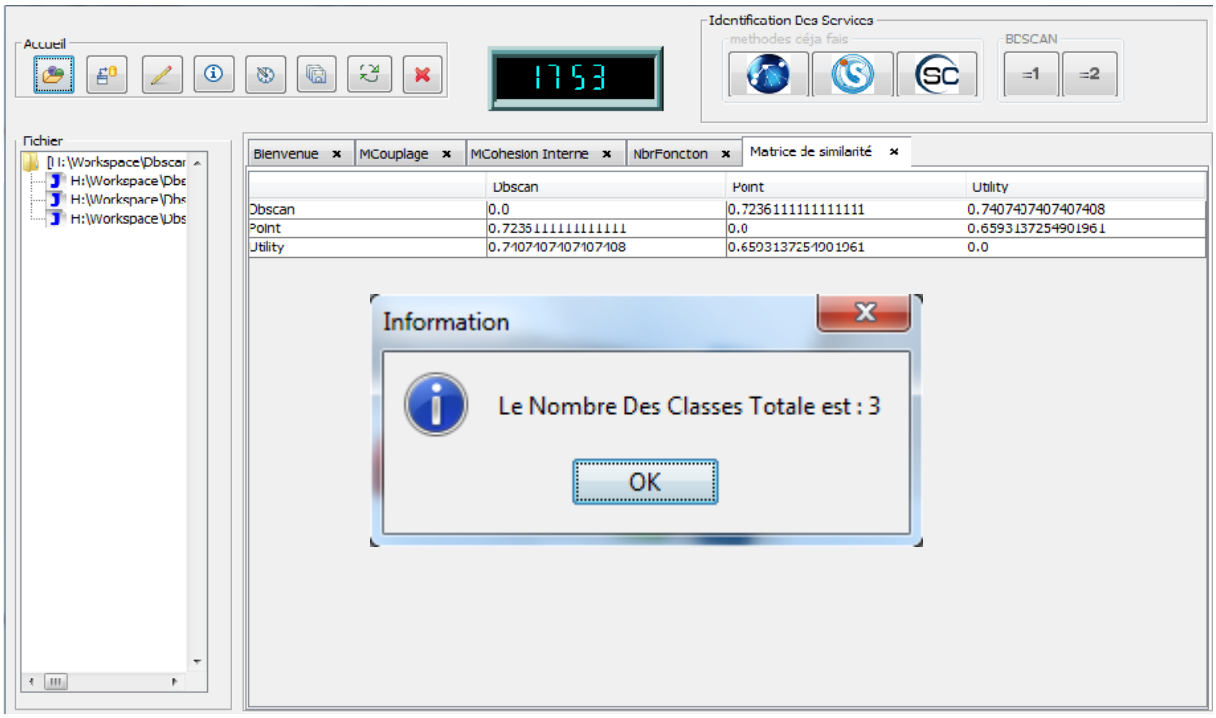


Figure 4.5 : *Fenêtre des résultats d'analyse d'un projet java.*

4.1.2 Récupérer une matrice de similarité à partir d'un fichier texte :

Cette option donne à l'utilisateur la possibilité de récupérer une matrice carré symétrique de similarité écrite dans un fichier texte (figure 4.6).

Bienvenue × matrice de similarite ×

	Classe0	Classe1	Classe2	Classe3	Classe4	Classe5	Classe6	Classe7	Classe8
Classe0	0.0	0.87084520...	0.74561403...	0.84722222...	0.88888888...	0.92753623...	0.92063492...	0.90196078...	0.88888888..
Classe1	0.87084520...	0.0	0.94595616...	0.72063917...	0.94117647...	0.94588744...	0.91904761...	0.95833333...	0.95238095..
Classe2	0.74561403...	0.94595616...	0.0	0.76944444...	0.96296296...	0.83333333...	1.0	1.0	1.0
Classe3	0.84722222...	0.72063917...	0.76944444...	0.0	0.83333333...	0.89473684...	0.88235294...	0.84615384...	0.81818181..
Classe4	0.88888888...	0.94117647...	0.96296296...	0.83333333...	0.0	0.97435897...	0.96969696...	0.95238095...	0.93333333..
Classe5	0.92753623...	0.94588744...	0.83333333...	0.89473684...	0.97435897...	0.0	1.0	1.0	1.0
Classe6	0.92063492...	0.91904761...	1.0	0.88235294...	0.96969696...	1.0	0.0	0.86904761...	1.0
Classe7	0.90196078...	0.95833333...	1.0	0.84615384...	0.95238095...	1.0	0.86904761...	0.0	0.83333333..
Classe8	0.88888888...	0.95238095...	1.0	0.81818181...	0.93333333...	1.0	1.0	0.83333333...	0.0
Classe9	0.91666666...	0.90909090...	0.97777777...	0.88333333...	0.86996336...	0.98245614...	0.98039215...	0.97435897...	0.96969696..
Classe10	0.94047619...	0.97530864...	1.0	0.91666666...	0.98148148...	1.0	1.0	1.0	1.0
Classe11	0.88888888...	0.86368393...	1.0	0.81818181...	0.93333333...	1.0	1.0	1.0	1.0
Classe12	0.76812739...	0.86171710...	0.91021825...	0.81147435...	0.9	0.93333333...	0.92753623...	0.91228070...	0.90196078..
Classe13	0.90277777...	0.94202898...	0.95555555...	0.86666666...	0.92857142...	0.96491228...	0.96078431...	0.94871794...	0.93939393..
Classe14	0.93548387...	0.89722222...	0.98484848...	0.91358024...	0.96825396...	0.98717948...	0.87301587...	0.80935672...	0.98148148..
Classe15	0.90196078...	0.94132653...	1.0	0.84615384...	0.91071428...	1.0	1.0	1.0	1.0
Classe16	0.88888888...	0.95238095...	1.0	0.81818181...	0.76666666...	1.0	1.0	1.0	1.0

Figure 4.6 : *Récupération d'une matrice de similarité à partir d'un fichier texte.*

4.1.3 Remplir une matrice de similarité :

L'utilisateur fixera la dimension de la matrice, une fois que la dimension est fixée une matrice carrée s'affiche (figure4.7).

La matrice remplie doit être une matrice symétrique.

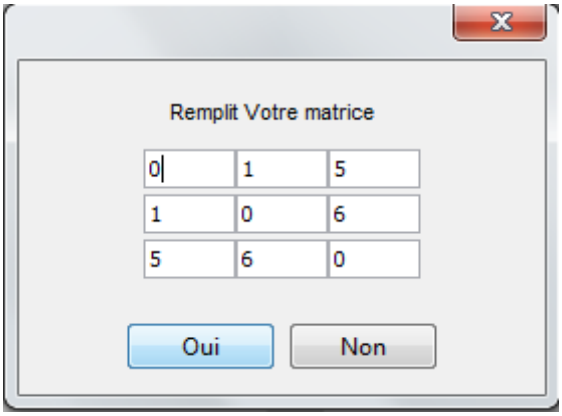


Figure 4.7 : *Matrice carrée de taille 3*3.*

4.1.4 Le guide d'utilisation :

Permet d'identifier dans quelques lignes notre projet globalement, ainsi il aide l'utilisateur à comprendre comment fonctionne notre application (figure 4.8).

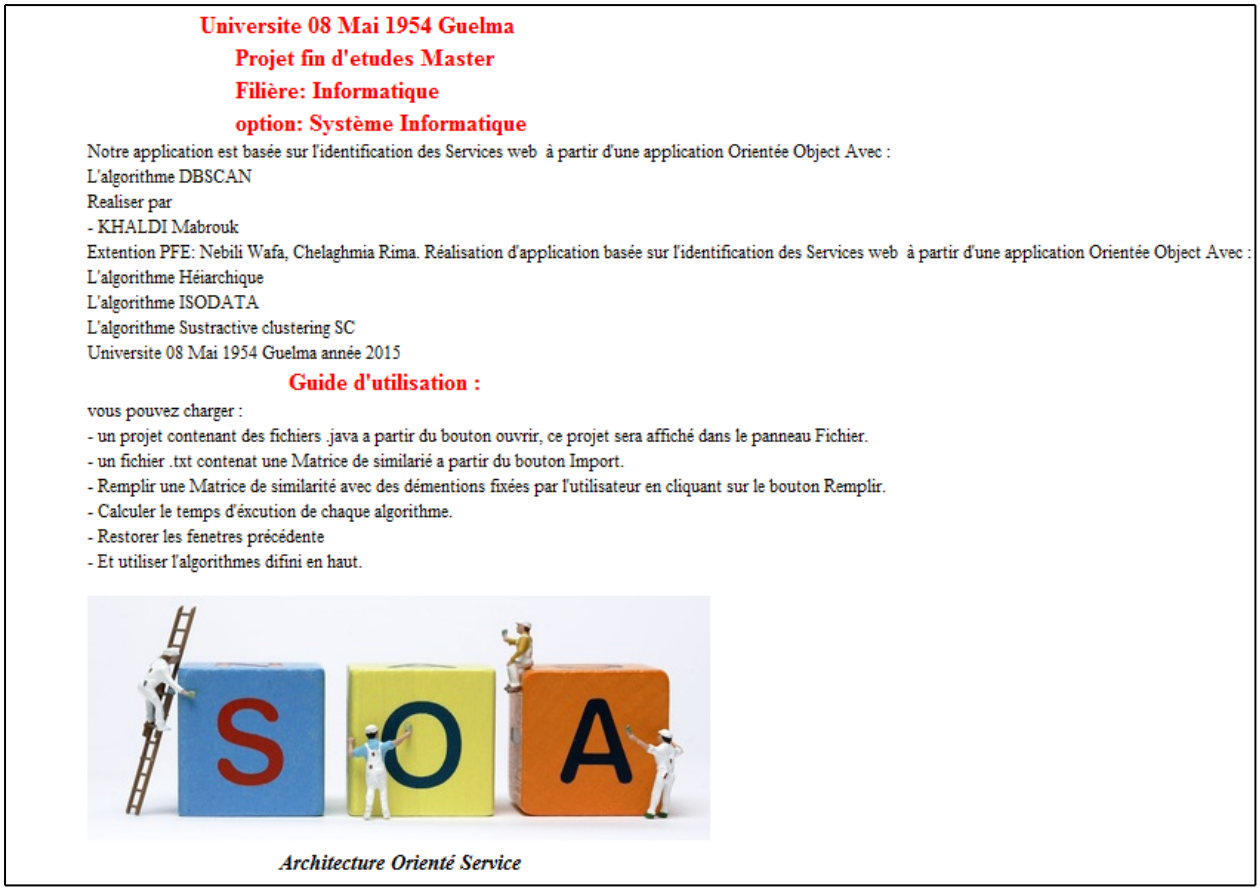


Figure 4.8 : *Le guide d'utilisation.*

4.1.5 Le calcul du temps d'exécution :

Après l'identification des services nous pouvons cliquer sur cette option pour voir le temps écoulé pendant l'exécution en nanoseconde (figure 4.9).

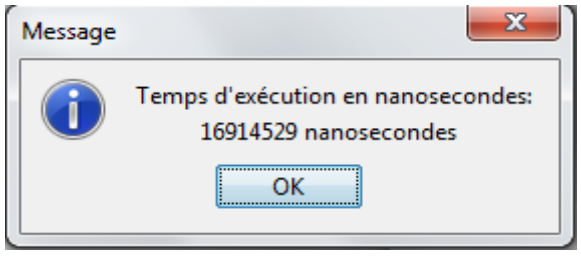


Figure 4.9: *Le temps d'exécution.*

4.1.6 Sauvegarder une matrice:

C'est une opération qui aide les utilisateurs à sauvegarder une matrice de similarité sous forme de texte dans n'importe quel endroit choisi par ce dernier, lorsque la matrice est sauvegardée une boite de dialogue sera affichée (figure 4.10).

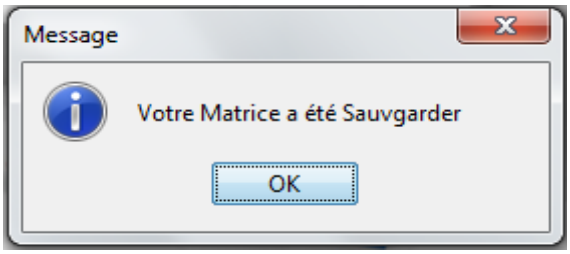


Figure 4.10: *Boite de confirmation de sauvegarde d'une matrice.*

4.1.7 Restaurer les fenêtres précédentes:

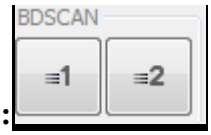
Ce bouton restaure les fenêtres quittées par l'utilisateur.

4.1.8 Quitter l'application:

Cette option permet d'abandonner l'application en cliquant sur ok pour confirmer la sortie.

4.2 Les algorithmes d'identification des services:

Après l'extraction de la matrice de similariténous pouvons choisir l'algorithme Dbscan.

DBSCAN algorithme : 

Nécessite comme paramètre d'entre Eps et MinPts dans notre cas Eps doit Etre entre [0 et 0.3] et le MinPts prend la valeur >= 3 par défaut comme il est montré à la figure 4.11

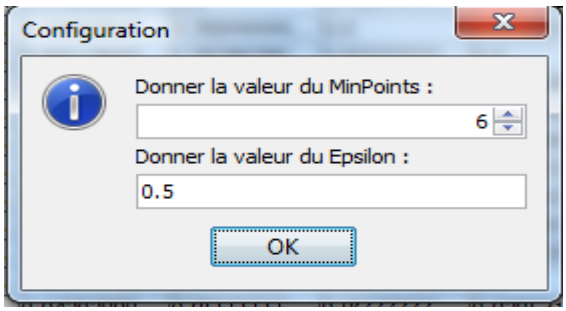


Figure 4.11: *Choix du MinPts et Eps pour DBSCAN*

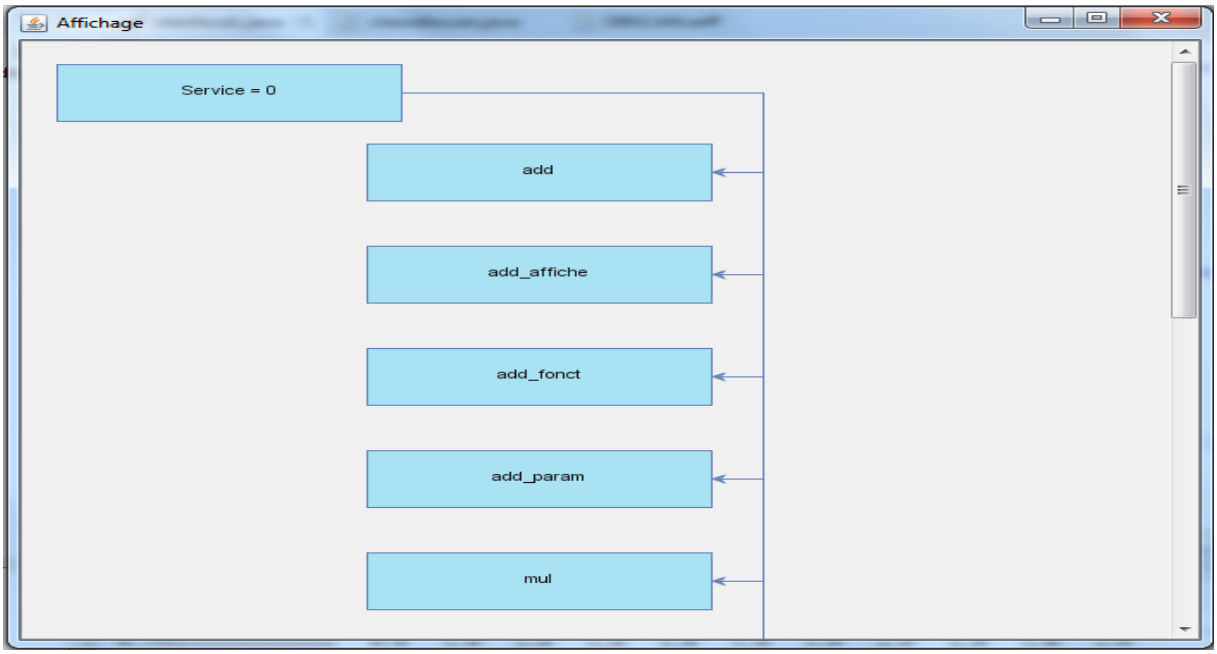


Figure 4.12: *Le résultat de L'algorithme DBSCAN sous forme graphique*

5 Expérimentation:

Afin de valider nos résultats nous avons choisi la démarche suivante :

Chercher dans les travaux similaires qui ont essayé de faire migrer des applications orientées objet vers la SOA. Nous avons prix 1 exemple afin de voir le comportement d'algorithme

5.1 Présentation des applications de test:

Nous avons choisi une application orientées objet comme exemple de test que nous connaissons à l'avance leur services résultant.

Exemple : ***Calcul_entier*** est une application simple qui contient 12 classes java, elle est composée de trois modules spécialisés dans le calcul sur deux entier (la somme, le produit et la soustraction). Chaque module est accessible à partir de son propre interface et utilise quelque fonctionnalités déclarées dans ces classes (fonction de lecture des entiers, fonction de calcul, et fonction d'affichage des résultats).

Le résultat attendu pour l'exemple ou Eps =0.3et MinPts =3 dans le tableau suivant :

L'exemple	Les résultats attendus	
	Le numéro de service	Les classes qui composent
Calcul_entier	Service= Noise	Add
		Add_affiche
		Add_fonct
		Add_param
		Mul
		Mul_affiche
		Mul_fnct
		Mul_param
		Sous
		Sous_affiche
		Sous_fnct

		Sous_param

Tableau 4.2 : *Les résultats attendus de l'exemple d'expérimentation.*

5.2 Le choix des paramètres :

5.2.1 Le choix d'Epset le MinPtsde la méthode (DBSCAN) adaptée:

L'application de cet algorithme pour l'exemple donnele tableau suivant :

Eps	MinPts=3	MinPts=4	MinPts=5
0.05	12	12	12
0.1	12	12	12
0.2	12	12	12
0.3	12	12	12
0.4	12	12	12
0.5	12	12	12

Tableau4.3 : *Le résultat de l'algorithme DBSCAN adaptée pour des différents Eps et différents MinPts.*

Nous remarquons que le choix d'Eps (epsilon c'est le rayon du noyau) et la valeur de MinPts (nombre minimum du cluster) donne une grande influence sur les résultats obtenues. La combinaison entre Eps et MinPts comme la classe avec la moyenne des similarités la plus petite donne un bon résultat et qui est le plus proche aux résultats attendus.

6Conclusion:

Dans ce dernier chapitre nous avons évalué notre algorithme (DBSCAN) suivant plusieurs valeurs d'Eps et MinPts afin de comparer les résultats. Notre évaluation montre que chaque l'algorithme a sa limite et ses avantages, et que le choix du meilleur algorithme dépend des besoins des utilisateurs.

Conclusion générale et perspectives

La réingénierie des systèmes patrimoniaux, notamment les systèmes orientés objet vers des systèmes orientés service est devenue un domaine de recherche d'actualité ; surtout avec les avantages apportés par la SOA. Nous nous sommes concentrés sur une étape essentielle, qui est l'identification de service. Le problème à été modalisé sous forme de problème de regroupement. Ainsi nous avons commencé à explorer les différents algorithmes de regroupement. Dans notre cas les groupes résultants ne sont pas connus en avance ; donc nous nous sommes concentrés sur les algorithmes de regroupement non supervisé. Le choix a été fait sur l'algorithme DBSCAN. Une phase d'adaptation a été nécessaire pour pouvoir l'appliquer à notre problématique ; à savoir le remplacement des distances par une formule de similarité qui utilise la notion de proximité basée sur le couplage et la cohésion entre classe ; puis le choix des paramètres qui conviennent à notre cas.

Le résultat du regroupement est un ensemble de classe-objet représentant des services candidats que l'utilisateur peut accepter ou réviser.

L'évaluation de la qualité de l'algorithme reste un problème ouvert, aucune méthode ne peut être qualifié d'universellement fiable.

Les résultats que nous avons étudié à travers plusieurs exemples, et à travers la comparaison avec des études similaires précédentes nous ont permis d'améliorer nettement l'adaptation et le paramétrage d'algorithme utilisé.

En guise de perspectives, nous envisageons d'axer les travaux futurs sur l'amélioration de l'algorithme, nous proposons de tester notre projet d'identification automatique de service sur des SI et des projets réels en cours d'exploitation.

Liste des webographies

[w1][http://adslbox.free.fr/rapports/rapport-gl-service-oriented-architecture.pdf24/05/2019]

[w2] [Didier Parigot , Baptiste Boussemartn LogNet INRIA Sophia Antipols - Méditerranée @004,route des Lucioles BP 93 F-06902 Sophia-Antipolis cedex, France Didier.Parigot@inria.fr, http://www-sop.inria.fr/lognet 2019]

[w3][http://adslbox.free.fr/rapports/rapport-gl-service-oriented-architecture.pdf 17/04/2019].

[w4][http://blog.xebia.fr/2009/03/04/soa-du-composant-au-service-le-contrat-standardise/,].

[w5][http://www.dotnet-france.com/Documents/WCF/Introduction%20a%20WCF.pdf, 17/05/2015].

[w6][http://www.redsen-consulting.com/2011/07/concepts-fondamentaux-soa].

[w7][http://genexo-a-m.googlecode.com/files/SOA%20final.doc31/12/2018].

[w8][http://www.journaldunet.com/developpeur/tutoriel/theo/05/10/13-explication-soa.shtml].

[w9][http://www.zdnet.fr/actualites/soa-comprendre-l-approche-orientee-service-39206712.htm14/05/2019].

[w10][http://download.docslide.fr/uploads/check_up03/212015/5560b541d8b42afe3b8b497a.pdf05/06/2015].

[w11][http://www.ledicodumarketing.fr/definitions/Reingenierie-des-Processus-de-Gestion.html10/06/2019].

[w12][http://tvquality.verollet.fr/files/Reingenierie_Rapport_Tornier_Verollet.pdf11/06/2019]

[w13][http://www.journaldunet.com/solutions/0403/040323_reengineering.shtml09/05/2019].

[w14][http://www.ricoh.fr/common-business-terms/business-process-re-engineering/14/05/2019].

[w15][http://hexawaretechnologies.fr/lmm-eng-3.htm 23/06/2019].

[w16][http://publicationslist.org/data/a.april/ref295/M%C3%A9moire%20Serge%20Bri%C3%A8re.pdf 21/03/2019].

[w17][http://www.iro.umontreal.ca/~dufour/cours/ift3912-h10/notes/19-Evolution.pdf,10/05/2019].

[w18] [http://dspace.univ-tlemcen.dz/bitstream/112/1045/4/Memoire.pdf 01/07/19]

[w19][http://apiacoa.org/publications/teaching/data-mining/clustering.pdf 05/04/2019].

[w20][http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20070038185.pdf 2019-06-09].

[w21][http://lipn.univ-paris13.fr/~cabanes/Publi/Nat/Cabanes_EGC08.pdf27/03/19]

[w22][http://fourier.eng.hmc.edu/e161/lectures/classification/node12.html 30/06/2019].

[w23][http://research.cs.tamu.edu/prism/lectures/pr/pr_l15.pdf 29/06/19].

[w24] [http://Clustering Approche par la théorie des jeux 29/06/19]

[w25][https://openclassrooms.com/fr/courses/4379436-explorez-vos-donnees-avec-des-algorithmes-non-supervises/4379571-partitionnez-vos-donnees-avec-dbscan05/07/2019].

[w26] [https://fr.wikipedia.org/wiki/Weka_(informatique)10/07/2019]

[w27] [Xu Zhang : Analyse de la similarité du code source pour la réutilisation automatique de tests unitaires à l'aide du CBR, université LAVAL, Québec, Canada, mémoire, 2019]

Liste des bibliographies

[1] Hüsemann Stefan, « SOA : L'utilité organisationnelle, technique et financière de l'architecture orientée service », Mémoire de magister, Université de Fribourg, Suisse, Août 2013.

[2] Brown A, Johnston S, Kelly K, « Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications », CA: Rational Software Corporation, Santa Clara, 2002.

[3]M.Nakamura, H.Igaki, T.Kimura, K.Matsumoto, « Extracting service candidates from procedural programs based on process dependency analysis », IEEE Asia-Pacific Services Computing Conference, p.484, 491, APSCC 2009, 7-11 Dec 2009.

[4]Livre blanc BEA, « SOA et virtualisation : quelle complémentarité », 2008.

[5] EJ.Chikofsky, JH.Cross II, « Reverse Engineering and Design Recovery: Taxonomy », IEEE Software, 7(1), p.13–17, Janvier 1990.

[6] Oracle, « Oracle IT Modernization Series: The Types of Modernization»**,** Oracle White Paper, September 2008.

[7] Séridi Ali, Seriai djamel-Abdelhak, Bourbia Riad, « Approches Pour L'Evolution Des Systèmes Patrimoniaux Vers Une Architecture Orientée Service », WOTIC 2011 : The Fourth Workshop on Information Technologies and Communication Casablanca, Morocco, Oct 13-15 2011.

[8] Sodki Chaari, « Interconnexion des processus Interentreprises: une approche orientée services », Thèse de doctorat, Lyon, 18 Décembre 2008.

[9] G.Lewis, E.Morris, D.Smith, L.Wrage, L.O'Brien, « Service-Oriented Migration and Reuse Technique (SMART) », Proceedings of 13th IEEE International Workshopon Software Technology and Engineering Practice (WSTEP'05), September 2005.

[10] Lotfi Khodja, « Contribution à la Classification Floue non Supervisée », thèse de doctorat, Université de Savoie, France.

[11] Kelaiaia Abdessalem, « Classification non supervisée de textes arabes appliquée à la recherche documentaire », Mémoire de magister, Université 08 mai 45 de Guelma, Algérie, 2007.

[12] Dawid Weiss, « Descriptive Clustering as a Method for Exploring Text Collections », PhD thesis, Institute of Computing Science Poznàn, Poland, 2006.

[13] Pavel Berkhin, « Survey of Clustering Data Mining Techniques», Accrue Software CA, USA, 2002.

[13] L.Candillier, « Contextualisation, visualisation et évaluation en apprentissage non supervisée », Thèse de doctorat, Université Charles De Gaulle Lille 3, France, 2006.

[20] A.K.Jain, M.N.Murty, P.J.Flynn, « Data clustering: a review », ACM Computer Survey, 31(3), p.264–323, 1999.

[14] Kelaiaia Abdessalem, « Classification non supervisée de textes arabes appliquée à la recherche documentaire », Mémoire de magister, Université 08 mai 45 de Guelma, Algérie, 2007.

[15] S.C.Johnson, «Hierarchical Clustering Schemes », Psychometrika, Vol.32, p.241-254, 1967.

[16] Ronald R.Yager, « S-mountain method for obtaining focus points from data ». International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 16(6), p.815–828, 2008.

[17] Nesma Settouti , « Renforcement de l'Apprentissage Structurel pour la reconnaissance du Diabète », Rapport de recherche, Université Abou Bekr Belkaid de Tlemcen, Algérie, 2011.

[18] Institute for Computer Science, University of Munich Oettingenstr. 67, D-80538 Miinchen, Germany ster I kriegel I sander I xwxu } @informatik.uni-muenchen.de

[19][http://bib.univoeb.dz:8080/jspui/bitstream/123456789/6755/1/m%C3%A9moire%20de%20kateb%20nabila.pdf]

[20][http://lim.univ-reunion.fr/staff/fred/M2info/1617/Stages/Rapports/Fanjanirina%20Rabetsivalaka_88595_assignsubmission_file_RapportFinal_Fanjanirina.pdf]

[21] Nebili wafa, Chelaghmiya rima université 08 mai 1945 Guelma projet fin d'étude 2015 master académiques L'identification des Services Web à partir d'une application Orientée Object Avec :L'algorithme Hiérarchique L'algorithme ISODATA L'algorithme Soustractive clustering SC

[22] Sylvain Chardign, « Extraction d'une architecture logicielle à base de composants depuis un système orienté objet », Thèse de doctorat, Université de Nantes, 23 octobre 2009.

[23] Lionel Briand, Prem Devanbu, Walcelio Melo, «An investigation into coupling measures for C++, In Proc of the Int », Conference on Software Engineering, ACM, p.412–421, 1997.

[24] Shyam Chidamber, Chris Kemerer, « A metrics suite for object-oriented design », IEEE Trans on Software Engineering, 20(6), p.476–493, juin 1994.

[25] Shyam Chidamber, Chris Kemerer, «Towards a metrics suite for object oriented design», Conference proceedings on Object-oriented programming systems, languages, and applications, p.197–211, ACM Press, New York, NY, USA, 1991.

[26] Martin Hitz, Behzad Montazeri, «Measuring coupling and cohesion in object-oriented systems. In Proc. Intl. Sym. on Applied Corporate Computing », Nov 1996.

[27] Y.Lee, B.Liang, S.Wu, F.Wan, «Measuring the coupling and cohesion of an object-oriented program based on information flow», ICSQ'95, p.81–90, 1995.

[28] W. Li, S.Henry, « Object oriented metrics that predict maintainability », Journal of Systems and Software, 223, p.111–122, 1993.

[29] Lazher Sadaoui , « Evaluation de la cohésion des classes : une nouvelle approche basé sur la classification »,Mémoire, Université du Québec, juin 2010.